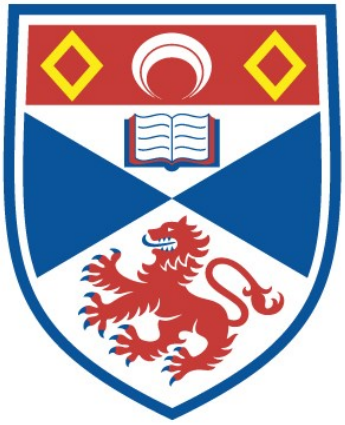
THE PREDICTION OF THERMAL PHASE-CHANGE BOUNDARIES AND ASSOCIATED TEMPERATURE DISTRIBUTIONS

A. S. Wood

A Thesis Submitted for the Degree of PhD
at the
University of St Andrews



1984

THE PREDICTION OF THERMAL PHASE-CHANGE BOUNDARIES AND ASSOCIATED TEMPERATURE DISTRIBUTIONS

A. S. Wood

Thesis submitted for the Degree of Doctor of Philosophy of the University of St. Andrews.

Th
19,001

ABSTRACT

The past three decades have seen a fast: expanding interest in thermal problems exhibiting a change of phase, more commonly known as Stefan problems. With the rapid advance in computer technology the use and expansion of numerical simulation schemes has been responsible for large advances in this field. The increasing size of computers has led to more sophisticated and complex numerical solutions becoming feasible from a computational point of view. On the other hand, part of this interest has developed from industrial quarters where a knowledge of the location of a melting/freezing boundary may be of critical importance for certain processes. Much experimental work has been completed in this field.

However, it is still useful to be able to obtain quick, accurate numerical solutions to such problems and it is with this in mind that this thesis is presented.

Ideas from both of the above areas of interest are treated. In the first case a simple to program and computationally efficient numerical scheme is proposed for solving one dimensional Stefan problems and its merits are discussed in relation to several of the more common existing solution schemes. This scheme is then modified to cater for a two dimensional problem which crudely imitates a possible heating configuration in some industrial processes. The problem, with its attendant difficulties, is first approximated by a 'test' problem which is constructed so as to admit an analytic solution. This allows assessment of the numerical procedure in two dimensions. In the course of this work a pseudo-analytic solution was obtained for the original two dimensional problem.

Finally, in collaboration with the British Gas Corporation, a

complex industrial freezing problem is discussed concerning the flow of liquid through an enclosed channel. Some simplifying assumptions are proposed to reduce the problem to a form for which a relatively simple numerical scheme may be adopted. Several simulations are completed to examine the effect of varying physical parameters on the solution and in particular to test for situations of blockage or steady-state.

DECLARATION

I declare that the following thesis is a record of research work carried out by me, that the thesis is my own composition, and that it has not been previously presented in application for a higher degree.

(Alastair Strang Wood)

POSTGRADUATE CAREER

I was admitted into the University of St. Andrews as a research student under Ordinance General No. 12 in October 1979 to work on The Prediction of Thermal Phase-Change Boundaries and Associated Temperature Distributions under the supervision of Dr. G. E. Bell. I was admitted under the above resolution as a candidate for the degree of Ph.D. in October 1980.

CERTIFICATE

I certify that Alastair S. Wood has satisfied the conditions of the Ordinance and Regulations and is qualified to submit the accompanying application for the Degree of Doctor of Philosophy.

DECLARATION

I declare that access to the thesis in the University Library shall be governed by any regulations approved by the Library Committee.

(Alastair Strang Wood)

ACKNOWLEDGMENTS

I would like to thank Dr. Graham Bell for his supervision and encouragement throughout the duration of my research and for providing helpful comments on the draft for this thesis.

I also wish to thank Dr. Mike Davies, Jeff Rhines and Dr. Barbara Lowesmith of the British Gas Corporation's Midlands Research Station for much useful discussion and suggestions during and since my visit to Solihull in the summer of 1981.

Finally, I wish to thank the Science and Engineering Research Council and the British Gas Corporation for providing financial support.

CONTENTS

CHAPTER I

A NOVEL APPROACH TO THE SOLUTION OF THE CLASSICAL STEFAN PROBLEM

1.1 Introduction

In the study of 'practical' melting and freezing problems, that is problems concerning at least two and preferably three space dimensions, the classical one-dimensional problem discussed by Stefan (1891) still plays a significant role [a more general result was given by Neumann (1912) in the 1860's]. It is generally accepted that thermal phase change problems modelling physical situations require the use of a numerical technique to obtain a solution. However, it may prove unwise to construct and apply a numerical algorithm to a problem without first obtaining some information concerning the likely behaviour of the algorithm since a 'well behaved' solution is not necessarily an accurate one. To this end the one-dimensional problem is useful in that it possesses a well documented analytic solution which may be used to examine the behaviour of a numerical algorithm by comparison [see Carslaw and Jaeger (1959), pp.285-291, for a detailed examination of the analytic results associated with this problem].

In this chapter a brief survey of several existing methods of solution is made including the use of some standard finite difference algorithms. Finally, a novel and efficient approach to the solution of the Stefan problem is proposed utilizing a solution method which has become known as the enthalpy method [Rose (1960), Szekely and Themelis (1971) ch.10, Atthey (1974)].

1.2 A Model Problem and Survey of Several Standard Solution Techniques

Literature has provided us with a plethora of techniques and numerical algorithms for solving the classical Stefan problem. Crank $(1975)_1$ gives a qualitative survey of many of these methods. In this section we present a brief quantitative comparison of several of the more common solution methods (using standard finite difference approximations) with a view to comparing them with a novel solution scheme to be presented in the following section.

For the purposes of illustration a simple model problem with a unique fusion temperature T_f is used. We consider a semi-infinite one-dimensional line, $x_1 \in [0,\infty)$, along which a material is initially solid at its fusion temperature. At time $t = 0.0$ heat energy is supplied at one end ($x_1 = 0$) in the form of a constant boundary temperature T_w ($>T_f$). The material subsequently melts and a liquid/solid interface propagates along the positive x_1-axis. In the liquid region the ensuing temperature distribution $T_L(x_1,t)$ is governed by the heat conduction equation

$$\frac{\partial T_L}{\partial t} = K_L \frac{\partial^2 T_L}{\partial x_1^2}, \quad 0 < x_1 < S(t), \; t > 0.0 \qquad - (1)$$

where K_L is the liquid thermal diffusivity (assumed constant) and $S(t)$ is the location of the interface. No heat conduction takes place in the solid region which remains at the fusion temperature. Noting the unique fusion temperature we have the boundary conditions

$$T_L(0,t) = T_w , \quad x_1 = 0.0, \quad t > 0.0$$

$$T_L(S(t),t) = T_f , \quad x_1 = S(t), \quad t > 0.0 \qquad - (2)$$

By virtue of a simple heat balance analysis at S(t) the location of the liquid/solid interface is governed by

$$L\rho \frac{dS}{dt} = - k_L \frac{\partial T_L}{\partial x_1} , \quad x_1 = S(t), \quad t > 0.0 \qquad - (3)$$

where L is the latent heat of fusion, ρ is the liquid density (assumed constant) and k_L is the liquid thermal conductivity (assumed constant). Initially there is no liquid region which implies the condition

$$S(0) = 0.0 , \quad t = 0.0 \qquad - (4)$$

Equations (1) and (3) subject to the conditions (2) and (4) respectively yield the following analytic solution [Carslaw and Jaeger (1959), p.285] for the liquid temperature distribution $T_L(x_1,t)$ and the interface location S(t).

$$T_L(x_1,t) = T_w + (T_f - T_w) \frac{\operatorname{erf}[x_1/2(\kappa_L t)^{\frac{1}{2}}]}{\operatorname{erf}[\lambda]} , \quad 0 \leqslant x_1 \leqslant S(t), \quad t \geqslant 0.0 \qquad - (5)$$

$$S(t) = 2\lambda(\kappa_L t)^{\frac{1}{2}} , \quad t \geqslant 0.0 \qquad - (6)$$

The melting/freezing parameter λ is, using equation (3) with the solutions (5) and (6), the solution of the transcendental equation

$$\frac{L\lambda\rho\pi^{\frac{1}{2}}}{(T_w-T_f)}\frac{\kappa_L}{k_L} = \frac{e^{-\lambda^2}}{erf[\lambda]}$$

It is convenient for the purposes of this illustration to non-dimensionalise the problem. Introducing the change of variables

$$x = \frac{x}{x_o} \quad , \quad U_L = \frac{(T_L-T_f)}{(T_w-T_f)} \quad , \quad \tau = \frac{k_L t}{\rho c x_o^2} \qquad - (7)$$

(where c is the specific heat capacity, x_o is a notional length and $\kappa_L = k_L/\rho c$) the problem is transformed to

$$\frac{\partial U_L}{\partial \tau} = \frac{\partial^2 U_L}{\partial x^2} \quad , \quad 0 < x < S(\tau), \ \tau > 0.0 \qquad - (8)$$

subject to

$$U_L(0,\tau) = 1.0 \quad , \quad x = 0, \ \tau > 0.0 \qquad - (9)$$

$$U_L(S(\tau),\tau) = 0.0 \quad , \quad x = S(\tau), \ \tau > 0.0$$

and

$$\alpha \frac{dS}{d\tau} = -\frac{\partial U_L}{\partial x}, \quad x = S(\tau), \ \tau > 0.0 \qquad - (10)$$

subject to

$$S(0) = 0.0 \ , \ \tau = 0.0 \qquad - (11)$$

where α , the latent heat parameter, is given by

$$\alpha = \frac{L\rho}{(T_w-T_f)}\frac{\kappa_L}{k_L} = \frac{L}{c(T_w-T_f)} \qquad - (12)$$

and is taken as <u>unity</u> in all the numerical algorithms (section 1.2) that follow. The classical solution becomes

$$U_L(x,\tau) = 1.0 - \frac{erf[x/2\tau^{\frac{1}{2}}]}{erf[\lambda]} \quad , \quad 0 \leqslant x \leqslant S(\tau), \ \tau \geqslant 0.0 \qquad - (13)$$

$$S(\tau) = 2\lambda\tau^{\frac{1}{2}}, \quad \tau \geq 0.0 \qquad - (14)$$

Equations (13) and (14) provide the analytic solutions with which to compare numerical solution schemes.

As a first approach to solving the problem defined by equations (8), (9), (10) and (11) we consider the standard implicit finite difference approximation. We construct a regular grid (with N intervals) of mesh size Δx over the line segment $0 \leq x \leq x_1$ where it is assumed that $S(\tau) < x_1$ for the range of times to be considered. To avoid a discontinuity in the initial temperature distribution we use the analytic solution (13) to initialize the numerical scheme at some time $\tau_0 > 0$. An implicit finite difference approximation to (8) and (9) is [Smith (1978), p.27]

$$U_i^{m+1} = U_i^m + r\,(U_{i+1}^{m+1} - 2U_i^{m+1} + U_{i-1}^{m+1}), \quad 1 \leq i \leq N-1, \; m = 0,1,2,\ldots$$

$$U_0^m = 1.0 \text{ and } U_N^m = 0.0 \qquad \forall\, m \qquad - (15)$$

where (the stability parameter) $r = \Delta\tau/(\Delta x)^2$ and $U_i^m \simeq U_L(i\Delta x, m\Delta\tau)$. At the point adjacent to the interface we require a modified finite difference approximation to (8). To maintain a similar accuracy to that of (15) (truncation error = $0(\Delta x^2)$) the following difference replacement is easily derived (remembering that the temperature is zero on the interface).

$$U_p^{m+1} = U_p^m + r\left(\frac{(\theta-1)}{(\theta+2)}U_{p-2}^{m+1} + \frac{2(2-\theta)}{(1+\theta)}U_{p-1}^{m+1} - \frac{(6+7\theta-\theta^3)}{\theta(1+\theta)(2+\theta)}U_p^{m+1}\right)$$

$$m = 0,1,2,\ldots \qquad - (16)$$

where θ is the fraction of the interval $[p\Delta x,(p+1)\Delta x]$ that is liquefied (see figure 1.1).

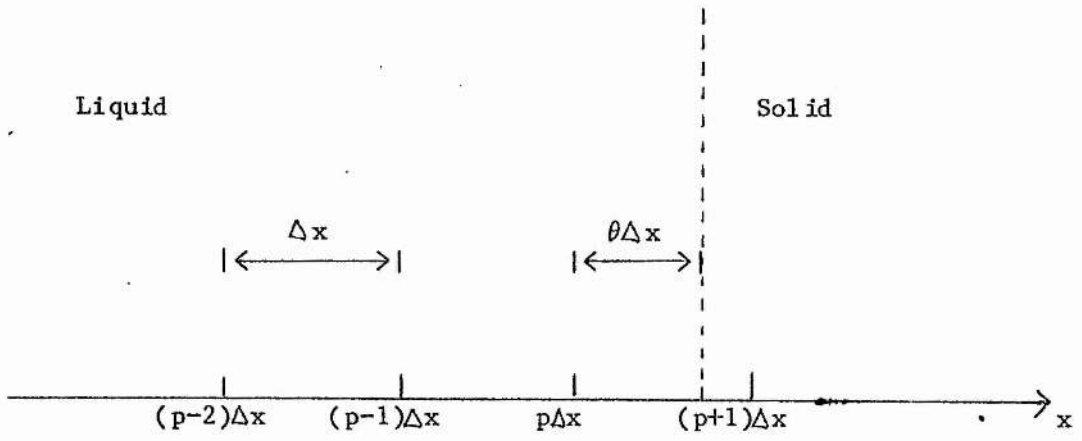


Figure 1.1

Grid point adjacent to the interface

From equation (10) we obtain

$$S^{m+1} = S^m - \Delta\zeta \left.\frac{\partial U}{\partial x}\right|_{x=S} , \quad m = 0,1,2,... \qquad - (17)$$

A suitable approximation to the temperature gradient in (17) is

$$\frac{1}{\Delta x}\left(\frac{-(1+\theta)U^m_p}{\theta} + \frac{\theta}{(1+\theta)} U^m_{p-1} \right) , \quad m = 0,1,2,...$$

A Gauss-Seidel scheme [Smith (1978), p.229] is used to solve (15) and (16) iteratively, and (17) is solved explicitly (see appendix 1.1 for the FORTRAN code). The results, which will be discussed in section 1.4, are presented in tables 1.1(a) and 1.1(b) for $\zeta = 1.0$ and $r = 0.4$ and in tables 1.2(a) and 1.2(b) for $\zeta = 1.0$ and $r = 2.0$. In all cases the numerical scheme was initialised at $\zeta_0 = 0.5$ using the analytic solution. The iterations are performed to eight figures. As

Table 1.1(a)

Values of the temperature distribution as predicted by the 'direct' implicit numerical scheme and the analytic solution at $\tau = 1.0$ for $r = 0.4$ (iterations to 8 figures)

	Implicit scheme			
x	$\Delta x=0.1$	$\Delta x=0.05$	$\Delta x=0.025$	Analytic solution
0.0	1.0	1.0	1.0	1.0
0.1	0.908933	0.908981	0.908994	0.908998
0.2	0.818322	0.818417	0.818441	0.818450
0.3	0.728616	0.728754	0.728789	0.728802
0.4	0.640250	0.640427	0.640472	0.640489
0.5	0.553236	0.553851	0.553904	0.553923
0.6	0.469179	0.469413	0.469473	0.469494
0.7	0.387222	0.387472	0.387536	0.387560
0.8	0.308094	0.308351	0.308417	0.308442
0.9	0.232078	0.232334	0.232400	0.232425
1.0	0.159416	0.159663	0.159728	0.159752
1.1	0.090308	0.090538	0.090599	0.090621
1.2	0.024907	0.025113	0.025168	0.025189
1.225	—	—	0.009403	0.009423

Table 1.1(b)

Predicted location of the interface at $\tau = 1.0$ for $r = 0.4$

Δx	$S(\tau)$	% Error
0.1	1.239692	-0.0349
0.05	1.240009	-0.0094
0.025	1.240094	-0.0025
Analytic Solution	1.240125	

Table 1.2(a)

Values of the temperature distribution as predicted by the 'direct' implicit numerical scheme and the analytic solution at $t = 1.0$ for $r = 2.0$ (iterations to 8 figures)

	Implicit scheme			
x	$\Delta x=0.1$	$\Delta x=0.05$	$\Delta x=0.025$	Analytic solution
0.0	1.0	1.0	1.0	1.0
0.1	0.908719	0.908925	0.908979	0.908998
0.2	0.817900	0.818306	0.818413	0.818450
0.3	0.728000	0.728592	0.728749	0.728802
0.4	0.639460	0.640219	0.640420	0.640489
0.5	0.552701	0.553601	0.553841	0.553923
0.6	0.468117	0.469130	0.469401	0.469494
0.7	0.386071	0.387163	0.387458	0.387560
0.8	0.306888	0.308024	0.308335	0.308442
0.9	0.230853	0.231997	0.232316	0.232425
1.0	0.158208	0.159325	0.159644	0.159752
1.1	0.089147	0.090206	0.090517	0.090621
1.2	0.023820	0.024794	0.025090	0.025189
1.225	--	--	0.009325	0.009423

Table 1.2(b)

Predicted location of the interface at $t = 1.0$ for $r = 2.0$

Δx	$S(t)$	% Error
0.1	1.237993	-0.1719
0.05	1.239505	-0.0500
0.025	1.239970	-0.0125
Analytic Solution	1.240125	

expected, with a refinement of the mesh size, the numerical solution displays convergence to the analytic solution.

To overcome the necessity of providing special difference replacements at the point adjacent to the interface Murray and Landis (1959) proposed keeping the number of intervals between x = 0.0 and x = S(τ) constant (equal to N, say) so that the interface always lies on a grid point (the N^{th}). By tracking particular grid lines (as opposed to constant x) and differentiating with respect to time they obtain for the i^{th} grid point

$$\left.\frac{\partial U}{\partial \tau}\right|_i = \left.\frac{\partial U}{\partial x}\right|_\tau \left.\cdot\frac{dx}{d\tau}\right|_i + \left.\frac{\partial U}{\partial \tau}\right|_x$$

By assuming the point x_i to move as

$$\frac{dx_i}{d\tau} = x_i \frac{1}{S(\tau)} \frac{dS}{d\tau}$$

equation (8) becomes

$$\left.\frac{\partial U}{\partial \tau}\right|_i = \frac{x_i}{S} \frac{dS}{d\tau} \cdot \frac{\partial U}{\partial x} + \frac{\partial^2 U}{\partial x^2} \qquad - (18)$$

with conditions $U_0 = 1.0$ and $U_N = 0.0$ for all τ. It should be noted that since there is a fixed number, N, of grid points then the mesh size $\Delta x = S(\tau)/N$ varies with time. An explicit numerical solution to this formulation is easily obtained. A suitable replacement for the temperature gradient at the interface ($x = N\Delta x$) is given by

$$\left.\frac{\partial U}{\partial x}\right|_{x=S} = \frac{(U^m_{N-2} - 4U^m_{N-1})}{2\Delta x} \quad , \quad m = 0,1,2,\ldots \qquad - (19)$$

Table 1.3(a)

Variable Space Grid: Values of the temperature distribution as predicted by the numerical (explicit finite difference) and analytic solutions at $\tau = 1.0$ (N is the number of space intervals)

	Explicit scheme				
x/S	N = 10	N = 20	N = 40	N = 80	Analytic solution
0.0	1.0	1.0	1.0	1.0	1.0
0.1	0.8871800	0.8871930	0.8871959	0.8871966	0.8871968
0.2	0.7752246	0.7752498	0.7752554	0.7752568	0.7752572
0.3	0.6649787	0.6650144	0.6650223	0.6650242	0.6650248
0.4	0.5572483	0.5572920	0.5573016	0.5573039	0.5573046
0.5	0.4527831	0.4528312	0.4528419	0.4528445	0.4528453
0.6	0.3522606	0.3523092	0.3523201	0.3523226	0.3523234
0.7	0.2562736	0.2563182	0.2563281	0.2563304	0.2563312
0.8	0.1653206	0.1653558	0.1653637	0.1653656	0.1653662
0.9	0.0797985	0.0798190	0.0798235	0.0798246	0.0798250
1.0	0.0	0.0	0.0	0.0	0.0

Table 1.3(b)

Predicted location of the interface at $\tau = 1.0$

N	S(τ)	% Error
10	1.240571	0.0340
20	1.240226	0.0081
30	1.240149	0.0019
40	1.240131	0.0005
Analytic Solution	1.240125	

and from (10) we may obtain a value for $\frac{dS}{d\tau}$ ($\equiv S'$). An explicit replacement for (18) yields

$$U_i^{m+1} = U_i^m + \frac{\Delta\tau x_i^m}{S^m} S' \left(\frac{U_{i+1}^m - U_{i-1}^m}{2\Delta x}\right) + r(U_{i+1}^m - 2U_i^m + U_{i-1}^m)$$

$$1 \leqslant i \leqslant N-1 \quad , \quad m = 0,1,2,\ldots$$

where $x_i^m \simeq i\Delta x^m$, Δx^m being the mesh size at the m^{th} time step. The new interface location may now be calculated from (17) and the new mesh size Δx^{m+1} from $\Delta x^{m+1} = S^{m+1}/N$. The numerical scheme was started using the analytic solution at $\tau_0 = 0.5$ and the results, discussed in section 1.4, are presented in tables 1.3(a) and 1.3(b) for $\tau = 1.0$ and a fixed time step (see appendix 1.2 for the FORTRAN code). The initial time step is such that $r \ll 0.5$. Since Δx subsequently increases then r decreases with time. It is seen that as the number of intervals is increased so the numerical solution converges to the analytic solution.

A logical extension to the method of Murray and Landis (1959) is to fix the Nth grid point (in space) by a suitable co-ordinate transformation, that is immobilize the interface [Landau (1950), Crank (1957), Ferriss (1975)]. By considering the new variable $\xi = x/S(\tau)$ the interface $x = S(\tau)$ is fixed at $\xi = 1.0$ in the co-ordinate system (ξ,τ). Utilizing elementary differential calculus the non-dimensional problem defined by (8), (9), (10) and (11) is transformed to the immobilized problem

$$\frac{\partial U}{\partial \tau} = \frac{\xi}{Z}\frac{dZ}{d\tau}\frac{\partial U}{\partial \xi} + \frac{1}{Z}\frac{\partial^2 U}{\partial \xi^2} \quad , \quad 0 < \xi < 1.0, \ \tau > 0.0 \qquad - (20)$$

$$U(0,\tau) = 1.0 \ , \ \xi = 0.0, \ \tau > 0.0 \qquad - (21)$$

Table 1.4(a)

A Boundary Immobilisation : Values of the temperature distribution as predicted by the numerical (explicit finite difference) and analytic solutions at $\tau = 1.0$ for $r = 0.4$

Immob Co-ord ξ	Explicit scheme $\Delta\xi=0.1$	$\Delta\xi=0.05$	$\Delta\xi=0.025$	$\Delta\xi=0.0125$	Analytic solution
0.0	1.0	1.0	1.0	1.0	1.0
0.1	0.887718	0.887193	0.887196	0.887197	0.887197
0.2	0.775225	0.775250	0.775255	0.775257	0.775257
0.3	0.664979	0.665014	0.665022	0.665024	0.665025
0.4	0.557248	0.557292	0.557302	0.557304	0.557305
0.5	0.452783	0.452831	0.452842	0.452844	0.452845
0.6	0.352261	0.352309	0.352320	0.352323	0.352323
0.7	0.256274	0.256318	0.256328	0.256330	0.256331
0.8	0.165321	0.165356	0.165364	0.165366	0.165366
0.9	0.079798	0.079819	0.079824	0.079825	0.079825
1.0	0.0	0.0	0.0	0.0	0.0

Table 1.4(b)

Predicted location of the interface at $\tau = 1.0$ for $r = 0.4$

$\Delta\xi$	$S(\tau)$	% Error
0.1	1.240302	0.0143
0.05	1.240159	0.0027
0.025	1.240133	0.0007
0.0125	1.240127	0.0000
Analytic Solution	1.240125	

$$U(1,\tau) = 0.0 \ , \ \xi = 1.0, \ \tau = 0.0$$

$$\frac{dZ}{d\tau} = - \frac{\partial U}{\partial \xi} \ , \ \xi = 1.0, \ \tau > 0.0 \qquad - (22)$$

$$Z(0) = 0.0 \ , \ \tau = 0.0 \qquad - (23)$$

To simplify the governing diffusion equation the interface location is redefined as $Z(\tau) = S^2(\tau)$. From (19) (with $\Delta x \rightarrow \Delta \xi$) and (22) we may obtain a value for $\frac{dZ}{d\tau}$. The temperature at the new time step may then be evaluated using the following explicit difference replacement of (20).

$$U_i^{m+1} = U_i^m + \frac{i\Delta\tau Z'}{2Z^m}(U_{i+1}^m - U_{i-1}^m) + \frac{r}{Z^m}(U_{i+1}^m - 2U_i^m + U_{i-1}^m) \qquad - (24)$$

$$1 \leqslant i \leqslant N-1 \ , \quad m = 0,1,2,\ldots$$

where $Z' = \frac{dZ}{d\tau}$ and $Z^m \simeq Z(m\Delta\tau)$. The numerical scheme was started at $\tau = 0.5$ using the analytic solution (13) and (14) (see appendix 1.3 for the FORTRAN code). A simple analysis of equation (24) implies that $r \leqslant Z^m/2$ for stability. The results, discussed in section 1.4, for $\tau = 1.0$ and $r = 0.4$ are presented in tables 1.4(a) and 1.4(b) and show the expected convergence to the analytic solution.

Another formulation that fixes the interface to lie on a grid point is the isotherm migration method [Chernous'ko (1970) and, independently, Dix and Cizek (1971)]. In this method x becomes the dependent variable, $x = x(U,\tau)$, and the solution gives the location x of a particular isotherm U at time τ. The interface location is easily obtained by ensuring that the fusion temperature is one of the grid points in the $U-\tau$ phase plane. Since

$$\frac{\partial U}{\partial x} = \left(\frac{\partial x}{\partial U}\right)^{-1} \quad , \quad \frac{\partial x}{\partial \tau} = -\frac{\partial U}{\partial \tau}\frac{\partial x}{\partial U}$$

and hence

$$\frac{\partial^2 U}{\partial x^2} = \left(\frac{\partial x}{\partial U}\right)^{-3}\frac{\partial^2 x}{\partial U^2}$$

then the problem of (8), (9), (10) and (11) becomes

$$\frac{\partial x}{\partial \tau} = \left(\frac{\partial x}{\partial U}\right)^{-2}\frac{\partial^2 x}{\partial U^2}, \quad 0 < U < 1.0, \ \tau > 0.0 \qquad - (25)$$

$$x(0,\tau) = S(\tau), \quad u = 0.0, \ \tau > 0.0 \qquad - (26)$$

$$x(1,\tau) = 0.0 \quad , \quad u = 1.0, \ \tau > 0.0$$

$$\frac{dS}{d\tau} = -\left(\frac{\partial x}{\partial U}\right)^{-1}, \quad u = 0.0, \ \tau > 0.0 \qquad - (27)$$

$$S(0) = 0.0 \ , \ \tau = 0.0 \qquad - (28)$$

An explicit difference replacement of N points for the problem yields

$$X_i^{m+1} = X_i^m + 4\Delta\tau\frac{(X_{i+1}^m - 2X_i^m + X_{i-1}^m)}{(X_{i+1}^m - X_{i-1}^m)^2} \quad , \quad 1 \leqslant i \leqslant N-1 \ , \ m = 0,1,2,\ldots \qquad - (29)$$

$$X_0^m = S^m \quad , \quad m = 0,1,2,\ldots$$

$$X_N^m = 0 \quad , \quad m = 0,1,2,\ldots$$

$$S^{m+1} = S^m - \frac{2\Delta U\Delta\tau}{(-3X_0^m + 4X_1^m - X_2^m)} \quad , \quad m = 0,1,2,.. \qquad - (30)$$

where $X_i^m \simeq X(i\Delta U, m\Delta\tau)$ and $S^m \simeq S(m\Delta\tau)$. To ensure that the isotherms (and interface) move in the correct direction (from physical considerations) the co-efficient of X_i^m in equation (29) must be

Table 1.5(a)

Isotherm Migration Method : Location of particular isotherms as predicted by the numerical scheme (explict finite difference) and the analytic solution at $\tau = 1.0$

Isotherm U	Explicit numerical technique				Analytic solution
	$\Delta U=0.1$	$\Delta U=0.05$	$\Delta U=0.025$	$\Delta U=0.0125$	
0.0	1.242759	1.240855	1.240319	1.240175	1.240125
0.1	1.088254	1.086707	1.086274	1.086158	1.086118
0.2	0.945778	0.944512	0.944160	0.944066	0.944033
0.3	0.812326	0.811291	0.811004	0.810927	0.810901
0.4	0.685757	0.684917	0.684685	0.684623	0.684602
0.5	0.564468	0.563798	0.563613	0.563565	0.563548
0.6	0.447204	0.446686	0.446544	0.446506	0.446493
0.7	0.332944	0.332566	0.332462	0.332435	0.332425
0.8	0.220822	0.220574	0.220507	0.220489	0.220483
0.9	0.110075	0.109952	0.109919	0.109910	0.109907
1.0	0.0	0.0	0.0	0.0	0.0

Table 1.5(b)

Predicted location of the interface at $\tau = 1.0$

ΔU	$S(\tau)$	% Error
0.1	1.242759	0.2124
0.05	1.240855	0.0589
0.025	1.240319	0.0156
0.0125	1.240175	0.0040
Analytic Solution		1.240125

positive and we impose the restriction [Dix and Cizek (1971)] at each time step

$$\Delta\tau < \frac{(X^m_{i+1} - X^m_{i-1})^2}{8}$$

for all i which implies that the time step varies as the scheme advances in time. The above formulation has been applied to the problem of a melting block of ice by Crank and Phahle (1973). To compute the above solution the time step is first calculated as

$$\Delta\tau = \min_{1\leqslant i\leqslant N-1} \left\{ \frac{(X^m_{i+1} - X^m_{i-1})^2}{8} \right\} \qquad - (31)$$

From (30) the new interface location S^{m+1} ($\equiv X_0^{m+1}$) is calculated and then (29) provides the remaining isotherm locations (see appendix 1.4 for the FORTRAN code). The results, discussed in section 1.4, are presented in tables 1.5(a) and 1.5(b) for $\tau = 1.0$ ($\tau_0 = 0.5$).

1.3 An Efficient Implementation of the Enthalpy Method

All of the foregoing solution schemes for the Stefan problem given by (8), (9), (10) and (11) work wholly in terms of the temperature which necessitates explicitly tracking the location of the liquid/solid interface. In terms of difference computations this either requires a modified difference replacement near the interface (see the "direct" implicit formulation) or a more complicated set of equations are produced (see the variable grid, boundary immobilization and isotherm migration method). We now give an account of a novel application of a technique which has become known as the enthalpy

method. Some of the work described in this section has appeared in the International Journal for Numerical Methods in Engineering [Wood, Ritchie and Bell (1981)].

It is convenient to define a total heat or 'enthalpy' function H(T) [Rose (1960), Szekely and Themelis (1971), Atthey (1974)] which is a function of temperature and combines the sensible and latent heats of the material. The requirement of working entirely in terms of the temperature is removed as is the necessity of explicitly tracking the interface. As a result of this, standard finite difference schemes may be applied over the whole solution region with due regard being taken of any stability requirements associated with a particular scheme. By considering the definition of the enthalpy function H(T) it is a straightforward exercise to extract the interface location from the solution at any time. The flexibility of the method has been demonstrated by its simple application to a problem involving a 'mushy' region in which the material melts or freezes over a temperature range [Voller and Cross (1981)].

For the model problem previously defined and following Atthey (1974) we have the following definition for the enthalpy function.

$$H_S \leqslant H \leqslant H_L \ , \quad T = T_f$$

$$H = \int_0^T c \, d\upsilon + L \ , \quad T > T_f$$

where $H_S = \int_0^{T_f} c \, d\upsilon$, $H_L = \int_0^{T_f} c \, d\upsilon + L$ and c is the specific heat capacity of the material. These relations may be re-arranged to yield

$$T = T_f \ , \quad cT_f \leqslant H \leqslant cT_f + L \qquad - (32)$$

$$T = \frac{(H - L)}{c} , \quad H > cT_f + L$$

Utilizing the non-dimensional change of variable

$$E = \frac{H - cT_f}{c(T_w - T_f)}$$

with those given by equations (7), then (32) becomes

$$U = 0.0 \quad , \quad 0 \leqslant E \leqslant \alpha \qquad - (33)$$
$$U = E - \alpha \, , \quad E > \alpha$$

where the latent heat parameter α is given by (12). The non-dimensional enthalpy formulation is then ($\alpha = 1.0$)

$$\frac{\partial E}{\partial \tau} = \frac{\partial^2 U}{\partial x^2} , \quad 0 < x < x_1 , \tau > 0.0 \qquad - (34)$$

$$U(x,0) = 0.0 , \quad x > 0.0, \tau = 0.0$$
$$U(0,\tau) = 1.0 , \quad x = 0.0, \tau > 0.0 \qquad - (35)$$
$$U(x_1,\tau) = 0.0 , \quad x = x_1, \tau > 0.0$$

with E(U) and $U(x,\tau)$ related by (33). Standard difference schemes may now be applied to the whole solution region.

Despite the computational simplicity of explicit difference schemes a drawback when solving, for example, the heat conduction problem is the stability restriction ($r (=\Delta\tau/(\Delta x)^2) \leqslant 1/2$ for the one-dimensional solution) which limits the size of the time step for a given mesh size. With implicit difference replacements this restriction is theoretically lifted. However, it then becomes necessary to solve a system of equations at each time step. A method

which combines both the computational simplicity of the explicit scheme and the large parameter range of the implicit scheme was proposed by Gourlay (1970). This scheme is given the name 'Hopscotch' because of the way in which the solution progresses through the space-time phase plane.

The odd-even hopscotch algorithm combines explicit and implicit finite difference approximations at alternate mesh points in such a way that the overall scheme does not require the solution of a system of equations, that is it is overall explicit.

We construct a uniform grid (of N points) of mesh size Δx over the region $0 \leqslant x \leqslant x_1$. The explicit odd-even difference replacement for equation (34) is

$$E_i^{m+1} = E_i^m + r(U_{i+1}^m - 2U_i^m + U_{i-1}^m) \ , \quad m = 0,1,2,... \qquad - (36)$$

for points where $m + i$ is <u>EVEN</u> and where $E_i^m \simeq E(i\Delta x, m\Delta t)$ and $U_i^m \simeq U(i\Delta x, m\Delta t)$. When $m + i$ is <u>ODD</u> we use a three point implicit difference replacement which is infact explicit in the sense that the temperatures at the nodes $(i+1)\Delta x$ and $(i-1)\Delta x$ at the new time step have already been calculated using the explicit replacement (36). The following procedure is adopted. Initially a value $\bar{E}_i^{m+1}$ is calculated from

$$\bar{E}_i^{m+1} = E_i^m + r(U_{i+1}^{m+1} + U_{i-1}^{m+1}) \ , \quad m = 0,1,2,... \qquad - (37)$$

If $\bar{E}_i^{m+1} \leqslant \alpha$ then $E_i^{m+1} \leqslant \alpha$ and hence, from (33), $U_i^{m+1} = 0.0$. If however $\bar{E}_i^{m+1} > \alpha$ it may be deduced that $E_i^{m+1} > \alpha$ and

$$E_i^{m+1} = \frac{(\tilde{E}_i^{m+1} + 2r\alpha)}{(1+2r)} , \quad m = 0,1,2,... \qquad - (38)$$

From (33) the corresponding temperature U_i^{m+1} is given by

$$U_i^{m+1} = E_i^{m+1} - \alpha , \quad m = 0,1,2,...$$

Equations (37) and (38) may be seen to be equivalent to one application of the implicit difference replacement ($\alpha = 1.0$)

$$E_i^{m+1} = E_i^m + r(U_{i+1}^{m+1} - 2U_i^{m+1} + U_{i-1}^{m+1}) , \quad m = 0,1,2,...$$

The odd-even scheme is particularly sensitive to discontinuous initial data [Bell and Ritchie (1980)] and so the numerical algorithm is initialised at $\tau = 0.5$ using the analytic solution. The initial enthalpy distribution may be determined from application of (33). To estimate the initial value of E(U) at the point(s) adjacent to the interface we suppose that each grid point is located at the right hand side of a segment of length Δx. The starting value of E(U) at this point is given as a fraction of the segment that is liquefied at the starting time. The results, discussed in section 1.4, are presented (for the case $\alpha = 1.0$) at $\tau = 1.0$ for $r = 0.4$ (tables 1.6(a) and 1.6(b)) and for $r = 2.0$ (tables 1.7(a) and 1.7(b)). For both cases it is evident that the algorithm converges as the difference mesh is refined (see appendix 1.5 for the FORTRAN code).

From the enthalpy solution obtained it is possible to determine the location of the interface by reversing the procedure described earlier for estimating E(U) in the solid region. The only non-zero value of E_i^m in the solid region is that which is adjacent to the

Table 1.6(a)

Enthalpy Formulation ('RIGHT' definition) : Values of the temperature distribution as predicted by the numerical and analytic solutions at τ = 1.0 for r = 0.4

Explicit method Δx=0.05	x	Hopscotch technique Δx=0.05	Δx=0.025	Δx=0.0125	Analytic solution
1.0	0.0	1.0	1.0	1.0	1.0
0.909990	0.1	0.910050	0.909516	0.909261	0.909998
0.820413	0.2	0.820533	0.818479	0.818972	0.818450
0.731695	0.3	0.731879	0.730331	0.729578	0.728802
0.644259	0.4	0.644514	0.642502	0.641508	0.640489
0.558527	0.5	0.558864	0.556396	0.555174	0.553293
0.474928	0.6	0.475355	0.472386	0.470961	0.469494
0.393893	0.7	0.394409	0.390804	0.389225	0.387560
0.315808	0.8	0.316390	0.311970	0.310295	0.308442
0.240923	0.9	0.241517	0.236284	0.234462	0.232425
0.169213	1.0	0.169738	0.164269	0.161880	0.159752
0.100258	1.1	0.100622	0.096255	0.092341	0.090621
0.033194	1.2	0.033325	0.031630	0.026596	0.025189

Table 1.6(b)

Predicted location of the interface at τ = 1.0 (Hopscotch)

Δx	S(τ)	% Error
0.05	1.230576	-0.7700
0.025	1.234900	-0.4213
0.0125	1.237786	-0.1886
Analytic Solution		1.240125

Table 1.7(a)

Enthalpy Formulation ("RIGHT" definition) : Values of the temperature distribution as predicted by the numerical and analytic solutions at $\tau = 1.0$ for $r = 2.0$

Explicit method $\Delta x=0.05$	x	Hopscotch technique $\Delta x=0.05$	$\Delta x=0.025$	$\Delta x=0.0125$	Analytic solution
	0.0	1.0	1.0	1.0	1.0
	0.1	0.911779	0.910648	0.909686	0.908998
	0.2	0.824173	0.821750	0.819826	0.818450
	0.3	0.737687	0.733756	0.730869	0.728802
	0.4	0.652361	0.647106	0.643247	0.640489
Method	0.5	0.568777	0.562222	0.557376	0.553923
Unstable	0.6	0.487106	0.479485	0.473642	0.469494
	0.7	0.407489	0.399209	0.392400	0.387560
	0.8	0.329939	0.321584	0.313975	0.308442
	0.9	0.254316	0.246622	0.238697	0.232425
	1.0	0.185876	0.176137	0.166838	0.159752
	1.1	0.124912	0.109440	0.098347	0.090621
	1.2	0.061244	0.046207	0.032452	0.025189

Table 1.7(b)

Predicted location of the interface at $\tau = 1.0$ (Hopscotch)

Δx	$S(\tau)$	% Error
0.05	1.265715	2.0635
0.025	1.259068	1.5275
0.0125	1.247850	0.6229
Analytic Solution	1.240125	

interface. This value represents the fraction of the segment of size Δx that is liquefied. The estimates produced in this way are presented in tables 1.6(b) and 1.7(b) at $\zeta = 1.0$ for $r = 0.4$ and $r = 2.0$ respectively.

An alternative algorithm for estimating the enthalpy is given by Crank $(1975)_2$ in which each grid point is considered to be at the centre of a segment of length Δx. Thus, for the point $x = i\Delta x$, the interface moves through values from $(i-0.5)\Delta x$ to $(i+0.5)\Delta x$ as the enthalpy E_i^m moves through the values 0.0 to 1.0. The results, discussed in section 1.4, are presented in tables 1.8(a) and 1.8(b) and tables 1.9(a) and 1.9(b) respectively for the above values of the parameters and r. Again convergence is evident.

1.4 Discussion and Conclusions

Tables 1.1(a) and 1.1(b) to 1.9(a) and 1.9(b) display the numerical results (for the algorithms described in sections 1.2 and 1.3) for an initial time of $\zeta = 0.5$ and a final time of $\zeta = 1.0$. It is observed that all the results show convergence to the analytic solution as the difference mesh is refined.

For the formulations described in section 1.2 it is possible to obtain a guide to the stability restrictions. It is well known that the explicit difference approximation to (8) requires that the parameter r be less than or equal to 1/2 for stability. The implicit scheme has unconditional stability and may admit an increased value of r and hence an increased time step for a given difference mesh. This is illustrated in table 1.2(a) ($r = 2.0$). The estimation of the interface location (table 1.2(b)) is very good, the percentage error

Table 1.8(a)

Enthalpy Formulation ('CENTRE' definition) : Values of the temperature distribution as predicted by the numerical and analytic solutions at $\tau = 1.0$ for $r = 0.4$.

Explicit method $\Delta x=0.05$	x	Hopscotch technique $\Delta x=0.05$	$\Delta x=0.025$	$\Delta x=0.0125$	Analytic solution
1.0	0.0	1.0	1.0	1.0	1.0
0.909022	0.1	0.909089	0.909035	0.909015	0.908998
0.818499	0.2	0.818632	0.818524	0.818484	0.818450
0.728871	0.3	0.729068	0.728913	0.728854	0.728802
0.640552	0.4	0.640813	0.640640	0.640558	0.640489
0.553919	0.5	0.554253	0.554121	0.554011	0.553923
0.469330	0.6	0.469766	0.469750	0.469601	0.469494
0.387185	0.7	0.387778	0.387776	0.387682	0.387560
0.308016	0.8	0.308821	0.308757	0.308580	0.308442
0.232514	0.9	0.233519	0.232526	0.232607	0.232425
0.161311	1.0	0.162370	0.159458	0.160058	0.159752
0.094489	1.1	0.095323	0.090989	0.090963	0.090621
0.031065	1.2	0.031385	0.029093	0.024557	0.025189

Table 1.8(b)

Predicted location of the interface at $\tau = 1.0$ (Hopscotch)

Δx	$S(\tau)$	% Error
0.05	1.241246	0.0904
0.025	1.240888	0.0615
0.0125	1.240576	0.0364
Analytic Solution		1.240125

Table 1.9(a)

Enthalpy Formulation ('CENTRE' definition) : Values of the temperature distribution as predicted by the numerical and analytic solutions at τ = 1.0 for r = 2.0

Explicit method Δx=0.05	x	Hopscotch technique Δx=0.05	Δx=0.025	Δx=0.0125	Analytic solution
	0.0	1.0	1.0	1.0	1.0
	0.1	0.911357	0.909909	0.909434	0.908998
	0.2	0.823076	0.820279	0.819324	0.818450
	0.3	0.735942	0.731566	0.730120	0.728802
	0.4	0.650490	0.644200	0.642257	0.640489
Method	0.5	0.566491	0.558570	0.556153	0.553923
Unstable	0.6	0.484643	0.475108	0.472201	0.469494
	0.7	0.405008	0.394206	0.390763	0.387560
	0.8	0.327603	0.316236	0.312147	0.308442
	0.9	0.252289	0.241413	0.236590	0.232425
	1.0	0.178786	0.169680	0.164468	0.159752
	1.1	0.114748	0.100598	0.096247	0.090621
	1.2	0.060366	0.033319	0.031582	0.025189

Table 1.9(b)

Predicted location of the interface at τ = 1.0 (Hopscotch)

Δx	S(τ)	% Error
0.05	1.284251	3.5582
0.025	1.259432	1.5569
0.0125	1.249583	0.7627
Analytic Solution		1.240125

decreasing from -0.1719% (for $\Delta x = 0.1$) to -0.0125% (for $\Delta x = 0.025$). However, for the latter case 400 time steps use ~18 seconds of CPU time. It should also be noted that an algorithm for solving a set of simultaneous algebraic equations at each time step is required thus complicating the computational details.

The position with regard to CPU time is somewhat worse when the implicit scheme is implemented for $r = 0.4$ (tables 1.1(a) and 1.1(b)) although the temperature and interface predictions are excellent with the error in the interface location decreasing from -0.0349% ($\Delta x = 0.1$) to -0.0025% ($\Delta x = 0.025$). The temperature predictions also display a converging solution.

A more efficient algorithm is obtained using the 'variable grid' formulation. The parameter r would appear to have an upper limit of 1/2 to ensure stability. However, for $r = 0.4$ the CPU times are considerably less (0.13s for 10 intervals, 8.22s for 40 intervals) than those for the implicit scheme with a similar number of intervals (0.5s for $\Delta x = 0.1$, 33.15s for $\Delta x = 0.025$). The accuracy is also greatly improved; for the interface estimation (table 1.3(b)) the percentage error ranges from 0.034% for 10 intervals to 0.0019% for 40 intervals.

A further improvement in the CPU time and accuracy is obtained using the boundary immobilisation scheme (table 1.4(a)). For a fixed value of r (= 0.4) the CPU times are of the order of 60% of the corresponding times for the 'variable grid' scheme. The accuracy of the interface estimations (table 1.4(b)) is approximately 3 times better ranging from 0.0143% ($\Delta\xi = 0.1$) to 0.0007% ($\Delta\xi = 0.025$). Further improvements in the CPU time can be made. For the explicit difference approximation to (20) the value of r is governed by the

relation $r \leqslant S^2/2$ for stability, where $S(\tau)$ is the interface location, and hence as the scheme progresses in time the value of r (and hence $\Delta\tau$) may be increased. A slight drawback, as with the 'variable grid' formulation, is the need to solve a more complicated differential equation which increases the computational detail.

The isotherm migration method displays similar CPU times to the boundary immobilisation scheme. The explicit difference approximation to (25) and (27) again admits a variable time step (equation (31)). Initially $\Delta\tau$ is relatively small, even for a course grid ($\Delta\tau_0 \simeq 0.003$ for $\Delta U = 0.1 \Rightarrow r \simeq 0.3$), but as the solution progresses this disadvantage is overcome (for $r = 0.4$ the implicit scheme takes 125 time steps ($\Delta x = 0.1$) compared to 115 time steps ($\Delta U = 0.1$) taken by the isotherm migration method). Another advantage is the ease with which the interface location is extracted although accuracy, while still very good, is decreased (0.2124% for $\Delta U = 0.1$ to 0.0156% for $\Delta U = 0.025$). Since we are tracking isotherms it is also easy to cater for a melting range by defining the maximum and minimum fusion temperatures as grid points in the difference approximation.

The scheme considered in section 1.3 does not yield to a simple stability analysis since the governing equations contain two dependent variables, $U(x,\tau)$ and $E(U)$, which are related by a <u>discontinuous</u> relation (equation (33)). However, computational evidence would suggest that the stability characteristics are similar to those for the pure conduction problem. The explicit difference approximation to equation (34) yields a solution for $r = 0.4$ (tables 1.6(a) and 1.8(a)) and for $r = 0.5$ [Wood, Ritchie and Bell (1981)]. For values of $r > 0.5$ (specifically $r = 2.0$, tables 1.7(a) and 1.9(a)) the purely explicit scheme is unstable. The hopscotch difference approximation

produces the expected converging numerical solution for r = 0.4 (tables 1.6(a) and 1.8(a)). For r = 2.0 the solution still appears stable and a good (converging) approximation to the analytic solution (tables 1.7(a) and 1.9(a)) is obtained. There is little to choose between the different enthalpy definitions (right or centre) - a comparison of the two does not seem to show any consistency. While for r = 0.4 the centre definition produces better (converging) interface estimations (c.f. tables 1.6(b) and 1.8(b)), at the higher value of r = 2.0 the right definition produces the more accurate estimations (c.f. tables 1.7(b) and 1.9(b)). Also, while the overall accuracy is worse than the previous numerical schemes, the enthalpy/hopscotch algorithm does offer a remarkable saving in CPU time. For r = 0.4 the times are comparable to those for the other schemes (table 1.10)

Table 1.10

CPU Times (r ≃ 0.4)

Scheme	r	Mesh	$\Delta\tau$(s)	Inter--vals	Time steps	% Error	CPU/s
Implicit	0.4	Δx=0.05	0.001	17-24	500	-0.0094	4.84
	0.4	Δx=0.025	0.00025	35-49	2000	-0.0025	33.57
Var Grid	↓	↑	0.000625	20	800	0.0081	0.97
	↓	↑	0.00015625	40	3200	0.0019	8.22
B Immob.	0.4	$\Delta\xi$=0.05	0.001	20	500	0.0027	0.60
	0.4	$\Delta\xi$=0.025	0.00025	40	2000	0.0007	4.79
Iso Mig.	↑	ΔU=0.05	↑	20	460	0.0589	0.53
	↑	ΔU=0.025	↑	40	1840	0.0156	4.34
Enthalpy	0.4	Δx=0.05	0.001	17-24	500	0.0904	0.91
	0.4	Δx=0.025	0.00025	35-49	2000	0.0615	6.74

(↑ and ↓ imply variable increasing and decreasing respectively). In

the above note that a positive percentage error implies an over estimation. For r = 2.0 the CPU times for the enthalpy/hopscotch algorithm are considerably reduced (for Δx=0.05, 0.025 and 0.0125 the CPU times in seconds are 0.08s, 0.88s and 6.87s respectively). Thus, for a mesh size of Δx=0.0125 the scheme produces an acceptable estimation of the interface location (% error ≃ 0.6229, table 1.7(b)) while only using ~6.87 seconds of CPU time.

The temperatures given in tables 1.6(a), 1.7(a), 1.8(a) and 1.9(a) all display a smooth distribution which would indicate a reasonable solution (c.f. the analytic solution). However, the enthalpy scheme as described in section 1.3 shows a serious deviation from the analytic solution if the temperature behaviour at one particular point is considered [Voller and Cross (1981) show this behaviour, which takes the form of a step-like function, and propose a remedy using linear interpolation]. Bell (1982) provides an explanation of this unusual behaviour.

In conclusion we may say that the enthalpy formulation when used in conjunction with the hopscotch numerical algorithm gives us a computationally efficient scheme for solving the Stefan problem posed in section 1.2. While the error in estimating the interface location is worse than the other methods descibed the value obtained (c.f. r=2.0, Δx=0.0125) is still a good approximation which may be well suited to a variety of industrial problems.

CHAPTER II

THE 'ELEMENT' PROBLEM

A TWO-DIMENSIONAL STEFAN PROBLEM WITH A BOUNDARY SINGULARITY

2.1 Introduction

In many instances a drawback associated with one-dimensional Stefan problems is their inability to realistically model a practical situation. For a variety of industrial processes a two-dimensional model is the minimum requirement for a satisfactory approximation. In this chapter we describe an investigation into a two-dimensional Stefan problem which is characterised by a particular solution difficulty.

Pure heat conduction problems in two-dimensions may yield an analytic solution. For example, Dirichlet's problem on a rectangle or circle admits a directly derived series solution [Boyce and Diprima (1965), pp.505-509]. Another example is heat conduction in an infinite right-angled corner [Carslaw and Jaeger (1959), pp.171-175]. In rectangular cartesian co-ordinates (x,y say) this yields a product solution in which the corresponding one-dimensional solutions are multiplied together. We are restricted in these approaches in that the initial temperature distribution must be expressable as $f(x)g(y)$ and the boundary conditions are generally constant temperature, constant flux etcetera.

However, for two-dimensional Stefan problems, including steady-state, it is the rule that no analytic solutions in terms of standard functions are available, even for simple boundary conditions

and geometries. Thus numerical methods are essentially the means for obtaining a progressive solution and it is this class of solutions on which this investigation concentrates.

For small time solutions it is often possible to construct series approximations valid for t, the time, less than some determined value t_{crit}. For large time approximations asymptotic methods may be employed. With regard to a numerical solution (for all time) we note that in one dimension it is possible to utilize one of many 'tried and tested' solution schemes and be able to analyse the validity of the solution. This is not necessarily true in two dimensions.

In section 2.2 we describe the model problem with its associated difficulties. In section 2.3 a test problem is constructed which imitates the essential characteristics of the model problem and which also has an analytic solution. This allows us to assess the behaviour of numerical (especially finite difference) solutions. A numerical solution of this test problem is proposed and discussed (with results) in section 2.4. In section 2.5 the enthalpy/hopscotch approach is extended to the two-dimensional model problem and the results are discussed with due regard being taken of sections 2.3 and 2.4.

2.2 The 'Element' Problem

Consider the upper half-plane $-\infty < u^* < \infty$, $v^* \geqslant 0$. Suppose that this region is occupied by a material initially solid and at its fusion temperature T_f. The boundary $v^* = 0$ is insulated (no heat flow across it) except for a finite element $u_1^* \leqslant u^* \leqslant u_2^*$ which is instantaneously raised (time $t^* = 0$) to a temperature T_w ($> T_f$) which is subsequently maintained. Heat energy will be conducted into the

material which will subsequently melt with a solid/liquid interface moving out from the heating element. A heat balance equation at the interface governs its motion in space and time. There is no heat conduction in the solid and the liquid temperature distribution is governed by the two-dimensional heat conduction equation. This describes a melting problem. The results for freezing are analogous.

The problem is slightly simplified in its presentation, and no generalisation is lost, if we consider the heating element to be symmetrically placed about the v^*-axis, $-h \leqslant u^* \leqslant h$. Thus the whole region is symmetric about the v^*-axis and we need only consider the positive quarter plane with no heat flow across the v^*-axis (figure 2.1).

A suitable mathematical description of the problem follows. The liquid phase temperature is governed by the heat conduction equation

$$\frac{\partial T^*}{\partial t^*} = \kappa \left(\frac{\partial^2 T^*}{\partial u^{*2}} + \frac{\partial^2 T^*}{\partial v^{*2}} \right) \qquad - (1)$$

$$(0 \leqslant u^* \leqslant h,\ v^* > 0),\ (u^* > h,\ v^* \geqslant 0),\ t^* > 0$$

subject to the initial and boundary conditions

$$T^*(u^*, v^*, 0) = T_f \qquad - (2)$$

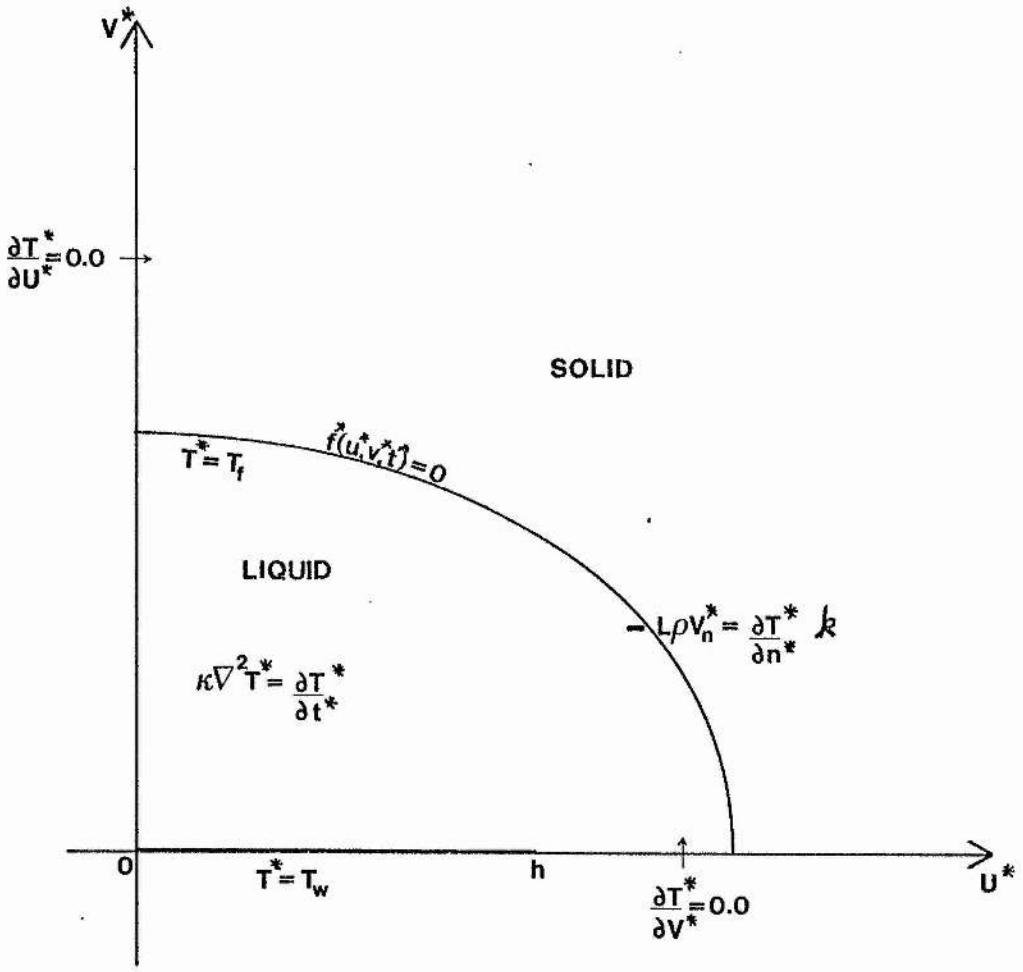
$$(0 \leqslant u^* \leqslant h,\ v^* > 0),\ (u^* > h,\ v^* \geqslant 0),\ t^* = 0$$

$$\frac{\partial T^*}{\partial u^*}(0, v^*, t^*) = 0\ , \quad u^* = 0,\ v^* > 0,\ t^* > 0 \qquad - (3)$$

$$T^*(u^*, 0, t^*) = T_w\ , \quad 0 \leqslant u^* \leqslant h,\ v^* = 0,\ t^* > 0 \qquad - (4)$$

Figure 2.1

The 'Element' Problem



$$\frac{\partial T^*}{\partial v^*}(u^*,0,t^*) = 0 \ , \quad u^* > h, \ v^* = 0, \ t^* > 0 \qquad - (5)$$

The heat balance condition at the interface requires that when the interface moves a distance dn^* (n^* being the outward normal to the interface at a point (u^*,v^*,t^*) on the interface) a quantity of heat $L\rho dn^*$ per unit area is absorbed and must be provided by conduction. If the interface curve is given by $f^*(u^*,v^*,t^*) = 0$ then this condition implies

$$L\rho V_n^* = - k \frac{\partial T^*}{\partial n^*} \ , \quad f^*(u^*,v^*,t^*) = 0; \ t^* > 0$$

where k is the thermal conductivity of the material, L is the latent heat of fusion and ρ is the solid density. V_n^* is the outward normal velocity of the interface. Following Patel (1968) we may re-arrange this equation into a more suitable form, namely

$$L\rho \frac{\partial f^*}{\partial t^*} = k \frac{\partial f^*}{\partial v^*} \left[1 + \left(\frac{\partial f^*/\partial u^*}{\partial f^*/\partial v^*}\right)^2\right] \frac{\partial T^*}{\partial v^*} \qquad - (6)$$

$$f^*(u^*,v^*,t^*) = 0, \ t^* > 0$$

Assuming that there is a unique fusion temperature we also have

$$T^*(u^*,v^*,t^*) = T_f \ , \quad f^*(u^*,v^*,t^*) = 0, \ t^* > 0 \qquad - (7)$$

Typically a physical problem admits a melting range which produces a 'solidus' and a 'liquidus' interface. The region between these

interfaces is usually known as the mushy region. It is not, as a rule, too difficult to incorporate this complication into a numerical model. However, for purposes of illustration we confine our investigation to the previously defined melting problem.

For the purpose of numerical calculation it is convenient to non-dimensionalise (and normalise) the problem. Introducing the variables

$$u = \frac{u^*}{h}, \quad v = \frac{v^*}{h}, \quad t = \frac{\kappa t^*}{h^2}, \quad T = \frac{T^* - T_f}{T_w - T_f}$$

the conduction problem, equations (1) to (7), becomes

$$\frac{\partial T}{\partial t} = \frac{\partial^2 T}{\partial u^2} + \frac{\partial^2 T}{\partial v^2}, \quad (0 \leqslant u \leqslant 1, v > 0), \ (u > 1, v \geqslant 0), \ t > 0 \qquad - (8)$$

subject to

$$T(u,v,0) = 0, \quad (0 \leqslant u \leqslant 1, v > 0), \ (u > 1, v \geqslant 0), \ t = 0 \qquad - (9)$$

$$\frac{\partial T}{\partial u}(0,v,t) = 0, \quad u = 0, \ v > 0, \ t > 0 \qquad - (10)$$

$$T(u,0,t) = 1, \quad 0 \leqslant u \leqslant 1, \ v = 0, \ t > 0 \qquad - (11)$$

$$\frac{\partial T}{\partial v}(u,0,t) = 0, \quad u > 1, \ v = 0, \ t > 0 \qquad - (12)$$

and

$$\frac{\partial f}{\partial t} = \kappa \frac{\partial f}{\partial v}\left[1 + \left(\frac{\partial f/\partial u}{\partial f/\partial v}\right)^2\right] \frac{\partial T}{\partial v}, \quad f(u,v,t) = 0, \ t > 0 \qquad - (13)$$

$$T(u,v,t) = 0, \quad f(u,v,t) = 0, \ t > 0 \qquad - (14)$$

κ is the latent heat parameter $c(T_w - T_f)/L$. The non-dimensionalised problem is illustrated in figure 2.2.

For purposes of computation equation (13) may not be in its most

Figure 2.2

The Non-dimensionalised 'Element' Problem

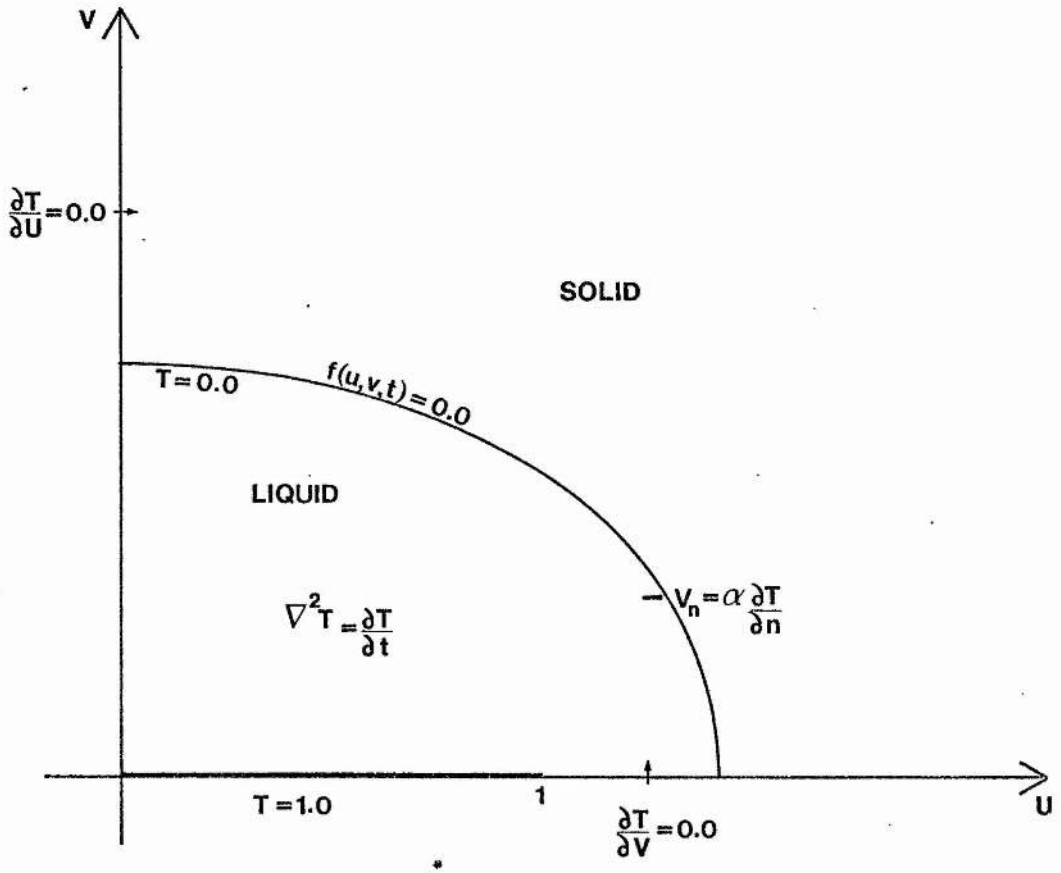
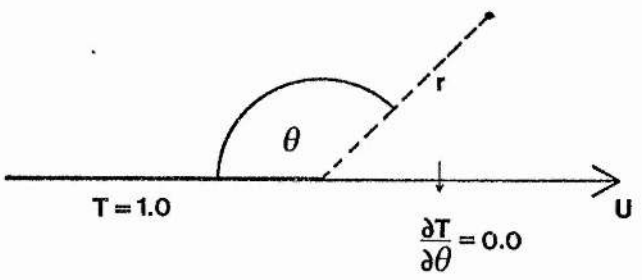


Figure 2.3

Local Co-ordinates for the Singular Point



convenient form. Suppose that the interface curve is given by $f(u,v,t) = v - s(u,t) = 0$, that is, the 'vertical' location of the interface is given as a function of u and t. By noting the differential identities

$$\frac{\partial f}{\partial t} = -\frac{\partial s}{\partial t}, \quad \frac{\partial f}{\partial u} = -\frac{\partial s}{\partial u}, \quad \frac{\partial f}{\partial v} = 1$$

an alternative presentation of (13) is

$$\frac{\partial s}{\partial t} = -\alpha\left[1 + \left(\frac{\partial s}{\partial u}\right)^2\right]\frac{\partial T}{\partial v}, \quad 0 \leqslant u \leqslant u_s,\ v = s(u,t),\ t > 0 \qquad -(15)$$

where u_s is the point of intersection of the interface and the u-axis. Thus (13) or (15) may be used for the heat balance equation according to convenience.

As mentioned earlier, difficulties may occur when attempting to solve this problem numerically, particularly using finite differences. The standard problem of discontinuous initial conditions is present. For small times the temperature close to the heating element may be approximated by a one-dimensional problem in v and t which has the solution

$$T \propto \operatorname{erf}(v/2\sqrt{t})$$

The heat flux F at any point in space and time is given by

$$F = -k\frac{\partial T}{\partial v} = -\frac{k\, e^{\frac{-v^2}{4t}}}{\sqrt{\pi t}}, \quad t > 0$$

We see that $F \to -k/\sqrt{\pi t}$ as $v \to 0$, that is, as we approach the element. It is clear that this expression is unbounded for $t \to 0$. This initial singularity is known to cause inaccurate numerical

solutions and so a small time starting solution would be useful.

Due to the sharp change in boundary condition at the point (1,0) there exists another singularity at the ends of the element. To illustrate the nature of this singularity, first recognised by Motz (1946), we consider a local series expansion about the point (1,0) [for example Bell and Crank (1973)]. With reference to figure 2.3 it can be shown that the local temperature behaviour is given by (see appendix 2.1)

$$\begin{aligned} T(r,\theta,t) = 1 &+ a_0(t)r^{\frac{1}{2}}\sin(\theta/2) + a_1(t)r^{\frac{3}{2}}\sin(3\theta/2) \\ &+ [a_2(t)\sin(5\theta/2) + \tfrac{1}{6}\dot{a}_0(t)\sin(\theta/2)]r^{\frac{5}{2}} \\ &+ [a_3(t)\sin(7\theta/2) + \tfrac{1}{10}\dot{a}_1(t)\sin(3\theta/2)]r^{\frac{7}{2}} + O(r^{\frac{9}{2}}) \end{aligned} \qquad - (16)$$

where the $a_i(t)$ are unknown functions of time that may be determined from remaining boundary conditions and $\dot{a}_i(t) \equiv \frac{da_i}{dt}(t)$. The heat flux F at a point (r,θ,t) contains a singular term. By differentiating (16) with respect to r it is clear that the second term in (16) will yield

$$\frac{a_0(t)\,\sin(\theta/2)}{2r^{\frac{1}{2}}}$$

which is obviously unbounded as r tends to zero ($a_0(t) \neq 0$). The accuracy of numerical solutions about this point becomes less reliable. As seen in the next section the results of a test problem would seem to indicate that estimates of the temperature about this point are high although the influence of the singularity appears to decrease at points further away.

2.3 A Test Problem

Much of the work in this section is included in a paper by Bell and Wood (1982) due to appear in the International Journal for Numerical Methods in Engineering.

The steady-state pure conduction equivalent of the problem described in the previous section may be solved by using a suitable conformal mapping [Churchill (1960), pp.191-195]. The transformed region admits simplified boundary conditions and because of the invariant nature of Laplace's equation the problem is effectively reduced to a one-dimensional situation. A closely related pseudo-steady-state problem has been studied by Siegel (1968) who investigated the freezing of a liquid flowing over a cold plate (a similar problem is also discussed by Goldstein and Siegel (1970)). The conformal mapping approach is extended to the continuous casting of a slab by Siegel (1978) in which a condition for the neglect of the time dependence of the heat conduction equation is provided. The transient heat conduction equation is not invariant but none the less the 'conformal transformation approach' provides us with some useful and interesting results.

In this section we begin by considering the non-dimensionalised problem described by equations (8) to (14). Suppose that the positive quadrant $u \geqslant 0$, $v \geqslant 0$ lies in the complex w-plane with $w = u + iv$ and $i^2 = -1$. Using the inverse of the mapping $w = \sin(z)$ the problem is mapped onto the region $0 \leqslant x \leqslant \pi/2$, $y \geqslant 0$ in the complex z-plane with $z = x + iy$ (figure 2.4). The mathematical description, leaving out the transformation details, becomes

Figure 2.4

The Transformed 'Element' Problem

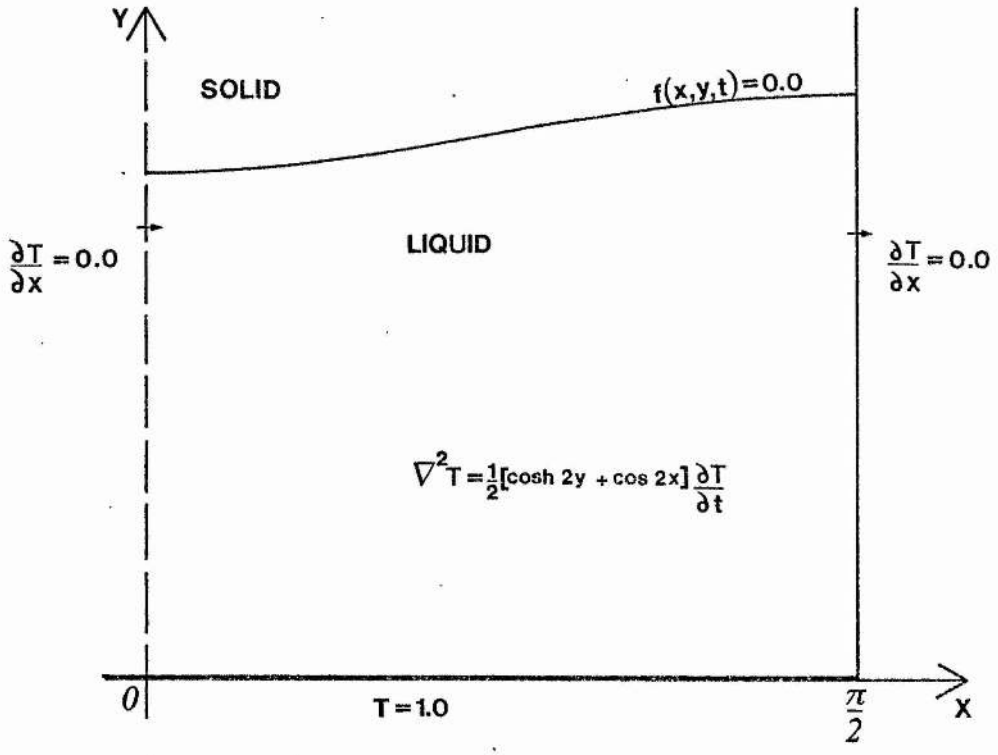
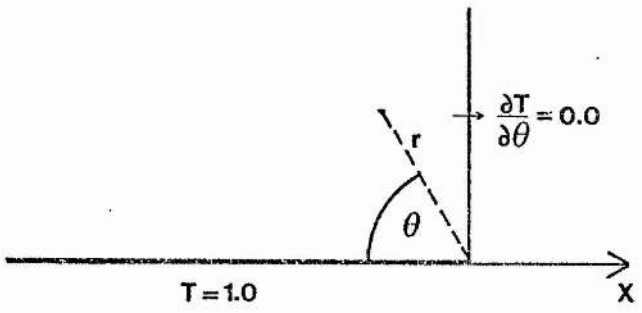


Figure 2.5

Local Co-ordinates for the Transformed Singular Point



$$\frac{1}{2}[\cosh(2y) + \cos(2x)] \frac{\partial T}{\partial t} = \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \qquad - (17)$$

$$0 \leqslant x \leqslant \frac{\pi}{2}, \; y > 0, \; t > 0$$

with conditions

$$T(x,y,0) = 0 , \quad 0 \leqslant x \leqslant \pi/2, \; y > 0, \; t = 0 \qquad - (18)$$

$$\frac{\partial T}{\partial x}(0,y,t) = 0 , \quad x = 0, \; y > 0, \; t > 0 \qquad - (19)$$

$$T(x,0,t) = 1 , \quad 0 \leqslant x \leqslant \pi/2, \; y = 0, \; t > 0 \qquad - (20)$$

$$\frac{\partial T}{\partial x}(\frac{\pi}{2},y,t) = 0 , \quad x = \frac{\pi}{2}, \; y > 0, \; t > 0 \qquad - (21)$$

and

$$\frac{1}{2}[\cosh(2y) + \cos(2x)] \frac{\partial f}{\partial t} = \alpha \frac{\partial f}{\partial y} \left[1 + \left(\frac{\partial f/\partial x}{\partial f/\partial y}\right)^2\right] \frac{\partial T}{\partial y} \qquad - (22)$$

$$f(x,y,t) = 0, \; t > 0$$

with attendant temperature condition

$$T(x,y,t) = 0 , \quad f(x,y,t) = 0, \; t > 0 \qquad - (23)$$

Using $f(x,y,t) = y - Y(x,t) = 0$, (22) becomes

$$\frac{1}{2}[\cosh(2y) + \cos(2x)] \frac{\partial Y}{\partial t} = -\alpha \left[1 + \left(\frac{\partial Y}{\partial x}\right)^2\right] \frac{\partial T}{\partial y} \qquad - (24)$$

$$0 \leqslant x \leqslant \frac{\pi}{2}, \; y = Y(x,t), \; t > 0$$

where $Y(x,0) = 0$ for all x.

The singular point at u = 1, v = 0 has been mapped to the point $x = \frac{\pi}{2}$, y = 0. With reference to figure 2.5 a local series expansion about the point $(\frac{\pi}{2},0)$ in the z-plane yields

$$T(r,\theta,t) = 1 + b_0(t)r\ \sin(\theta) + b_1(t)r^3\ \sin(3\theta) + O(r^5) \quad - (25)$$

and the corresponding heat flux F is given by

$$F(r,\theta,t) = - k\ [b_0(t)\sin(\theta) + 3b_1(t)r^2\ \sin(3\theta) + O(r^4)\ ] \quad - (26)$$

Clearly $F(r,\theta,t)$ is bounded as r -> 0 and the singular nature of the point (1,0) in the w-plane appears to have been removed. However, closer examination of the transformed conduction equation (17) reveals that the equation changes form as the point $(\frac{\pi}{2},0)$ is approached. In a finite difference approximation to (17) this leads to a singular co-efficient.

However, if we reverse the transformation procedure we can obtain some very useful results with respect to the numerical solution of the problem given by equations (8) to (15). We start by considering the standard non-dimensionalised two-dimensional Stefan problem (with melting) in the z-plane. That is, we consider the heat conduction problem defined over the region $0 \leqslant x \leqslant \frac{\pi}{2}$, $y \geqslant 0$ which contains a material initially at its fusion temperature, zero. At time t = 0 the element $0 \leqslant x \leqslant \frac{\pi}{2}$ (y = 0) is raised to unit temperature, the sides x = 0 and $x = \frac{\pi}{2}$ (y > 0) being insulated for all time. The material subsequently melts and a solid/liquid interface propagates in the y-direction. Mathematically we have

$$\frac{\partial T}{\partial t} = \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \quad , 0 \leqslant x \ll \frac{\pi}{2}, \; y > 0, \; t > 0 \qquad - (27)$$

subject to

$$T(x,y,0) = 0 \quad , 0 \leqslant x \ll \frac{\pi}{2}, \; y > 0, \; t = 0 \qquad - (28)$$

$$\frac{\partial T}{\partial x}(0,y,t) = 0 \quad , x = 0, \; y > 0, \; t > 0 \qquad - (29)$$

$$T(x,0,t) = 1 \quad , 0 \leqslant x \ll \frac{\pi}{2}, \; y = 0, \; t > 0 \qquad - (30)$$

$$\frac{\partial T}{\partial x}\left(\frac{\pi}{2},y,t\right) = 0 \quad , x = \frac{\pi}{2}, \; y > 0, \; t > 0 \qquad - (31)$$

The heat balance at the interface is

$$\frac{\partial Y}{\partial t} = -\alpha \frac{\partial T}{\partial y} \quad , 0 \leqslant x \ll \frac{\pi}{2}, \; y = Y(x,t), \; t > 0 \qquad - (32)$$

Due to the conditions (29) and (31) this problem is satisfied by the one-dimensional solution

$$T(x,y,t) = 1 - \frac{\operatorname{erf}(y/2\sqrt{t})}{\operatorname{erf}(\lambda)} \quad , 0 \leqslant x \ll \frac{\pi}{2}, \; y \geqslant 0, \; t \geqslant 0 \qquad - (33)$$

where λ satisfies

$$\frac{e^{-\lambda^2}}{\operatorname{erf}(\lambda)} = \frac{\lambda \pi^{\frac{1}{2}}}{\alpha}$$

and the interface $Y(x,t)$ is given by $Y(x,t) = 2\lambda t^{\frac{1}{2}}$ for $0 \leqslant x \ll \frac{\pi}{2}$, $t \geqslant 0$. Thus, the solid/liquid interface is a straight line for all time.

We now consider the conformal transformation $w = \sin(z)$ ($z,w \in \mathbb{C}$, $z = x + iy$, $w = u + iv$, $x,y,u,v \in \mathbb{R}$, $i^2 = -1$). It is easily shown that the relations between the co-ordinate systems are

$u = \sin(x)\cosh(y)$ and $v = \cos(x)\sinh(y)$. The governing equation (27) becomes

$$\frac{\partial T}{\partial t} = |\cos(z)|^2 \left(\frac{\partial^2 T}{\partial u^2} + \frac{\partial^2 T}{\partial v^2}\right)$$

where T is now a function of u, v and t. After some manipulation it is readily seen that

$$|\cos(z)|^2 = [(u+1)^2 + v^2]^{\frac{1}{2}} [(u-1)^2 + v^2]^{\frac{1}{2}}$$

and the above equations (27) to (32) are mapped to the following Stefan problem in the w-plane ($u \geqslant 0$, $v \geqslant 0$). For the temperature distribution we have

$$\frac{\partial T}{\partial t} = [(u+1)^2+v^2]^{\frac{1}{2}} [(u-1)^2+v^2]^{\frac{1}{2}} \left(\frac{\partial^2 T}{\partial u^2} + \frac{\partial^2 T}{\partial v^2}\right) \qquad - (34)$$

$$(0 \leqslant u \leqslant 1,\ v > 0),\ (u > 1,\ v \geqslant 0),\ t > 0$$

subject to

$$T(u,v,0) = 0 \quad , (0 \leqslant u \leqslant 1, v > 0),\ (u > 1, v \geqslant 0),\ t = 0 \qquad - (35)$$

$$\frac{\partial T}{\partial u}(0,v,t) = 0 \quad , u = 0,\ v > 0,\ t > 0 \qquad - (36)$$

$$T(u,0,t) = 1 \quad , 0 \leqslant u \leqslant 1,\ v = 0,\ t > 0 \qquad - (37)$$

$$\frac{\partial T}{\partial v}(u,0,t) = 0 \quad , u > 1,\ v = 0,\ t > 0 \qquad - (38)$$

For the interface we suppose that the curve $y = Y(x,t)$ is mapped to the curve $f(u,v,t) = 0$ in the w-plane. Noting that

$$\frac{\partial T}{\partial y}(x,Y(x,t),t) = |\cos(z)| \frac{\partial T}{\partial n}(u,v,t) \quad , f(u,v,t) = 0$$

(where n is the outward normal to the curve $f(u,v,t) = 0$) and, since the transformation multiplies lengths by the factor $|F'(z)| = |\cos(z)|$ ($F(z)$ is the mapping),

$$\frac{\partial n/\partial t}{\partial Y/\partial t} = |\cos(z)|$$

the interface condition is transformed to

$$\kappa |\cos(z)|^2 \frac{\partial T}{\partial n} = -\frac{\partial n}{\partial t} \quad , f(u,v,t) = 0, \; t > 0$$

This is equivalent to

$$\kappa[(u+1)^2+v^2]^{\frac{1}{2}}[(u-1)^2+v^2]^{\frac{1}{2}}\frac{\partial T}{\partial n} = -\frac{\partial n}{\partial t} \quad , f(u,v,t) = 0, \; t > 0 \qquad - (39)$$

Also at the melting front

$$T(u,v,t) = 0 \quad , f(u,v,t) = 0, \; t > 0 \qquad - (40)$$

Despite the fact that the condition (39) does not yield to any reasonable physical interpretation this test problem (equations (34) to (40)) still contains several points characteristic of the 'element' problem (equations (8) to (14)). First, if the discontinuous initial data is used there is the usual problem of poor numerical accuracy at the source of the 'shock'. Second, the point (1,0) is a singular point with a local temperature distribution given by the series solution

$$T(r,\theta,t) = 1 + a_0(t)r^{\frac{1}{2}}\sin(\theta/2) + a_1(t)r^{\frac{3}{2}}\sin(\theta/2) \qquad - (41)$$

$$+ a_2(t)r^{\frac{3}{2}} \sin(3\theta/2) + O(r^{\frac{5}{2}})$$

which clearly yields an unbounded heat flux as r -> 0. Finally, there is still the problem of tracking a moving boundary. However, one advantage is that the problem admits an analytic solution. The temperature distribution is merely the transformation of equation (33) while, by simple calcululation, the line Y(x,t) becomes the ellipse

$$f(u,v,t) = \frac{u^2}{\cosh^2 Y} + \frac{v^2}{\sinh^2 Y} - 1 = 0 \quad , \; t > 0 \qquad - (42)$$

From $u = \sin(x)\cosh(y)$, $v = \cos(x)\sinh(y)$ and (33) the temperature distribution is given by

$$T(u,v,t) = 1 - \frac{\operatorname{erf}(\phi(u,v)/2\sqrt{t})}{\operatorname{erf}(\lambda)} \quad , \; f(u,v,t) \leqslant 0, \; t \geqslant 0 \qquad - (43)$$

where

$$\phi(u,v) = \frac{1}{2} \cosh^{-1}\{(u^2+v^2) + ((u+1)^2+v^2)^{\frac{1}{2}}((u-1)^2+v^2)^{\frac{1}{2}}\}$$

Equations (42) and (43) allow a quantitative analysis of the numerical solution of equations (34) to (40).

2.4 A Numerical Solution of the Test Problem

This section describes a finite difference solution of the test problem given by equations (34) to (43). To alleviate the problem of numerically tracking the interface location we shall consider a two-dimensional enthalpy reformulation of the above problem. Denoting the enthalpy of a point (u,v,t) by E(u,v,t) equation (34) becomes

$$\frac{\partial E}{\partial t} = [(u+1)^2+v^2]^{\frac{1}{2}}\,[(u-1)^2+v^2]^{\frac{1}{2}}\,\left(\frac{\partial^2 T}{\partial u^2} + \frac{\partial^2 T}{\partial v^2}\right) \qquad - (44)$$

$$(0 \leqslant u \leqslant 1,\ v > 0),\ (u > 1,\ v \geqslant 0),\ t > 0$$

with the temperature T(u,v,t) subject to the conditions (35) to (38), and (40). The interface is catered for by defining the following relation between E(u,v,t) and T(u,v,t).

$$T(u,v,t) = E(u,v,t) - 1/\mathcal{K} \quad , \quad E(u,v,t) > 1/\mathcal{K} \qquad - (45)$$

$$= 0 \quad \text{otherwise}$$

For illustrative purposes we will consider the latent heat parameter $\mathcal{K}$ to be unity.

With regard to the numerical solution we construct a rectangular grid of mesh size $\Delta = \Delta u = \Delta v$ over the solution region. From the analytic solution we need only consider the region $u \in [0.0,2.0]$, $v \in [0.0,1.6]$. Using a backward difference formula for the time derivitive and central differences for the spacial derivitives the explicit finite difference approximation to equation (44) is

$$E^{m+1}_{i,j} = E^{m}_{i,j} + \varepsilon[(u_i+1)^2+v_j^2]^{\frac{1}{2}}\,[(u_i-1)^2+v_j^2]^{\frac{1}{2}} \qquad - (46)$$

$$\times\ (T^{m}_{i+1,j} + T^{m}_{i-1,j} + T^{m}_{i,j+1} + T^{m}_{i,j-1} - 4T^{m}_{i,j})$$

for $0 \leqslant i \leqslant N_u$ and $0 \leqslant j \leqslant N_v$. $E^{m}_{i,j} \simeq E(i\Delta, j\Delta, m\Delta t)$, $T^{m}_{i,j} \simeq T(i\Delta, j\Delta, m\Delta t)$, ε is the stability parameter $\Delta t/\Delta^2$ and N_u and N_v

Figure 2.6

Initial Interface Enthalpy Values

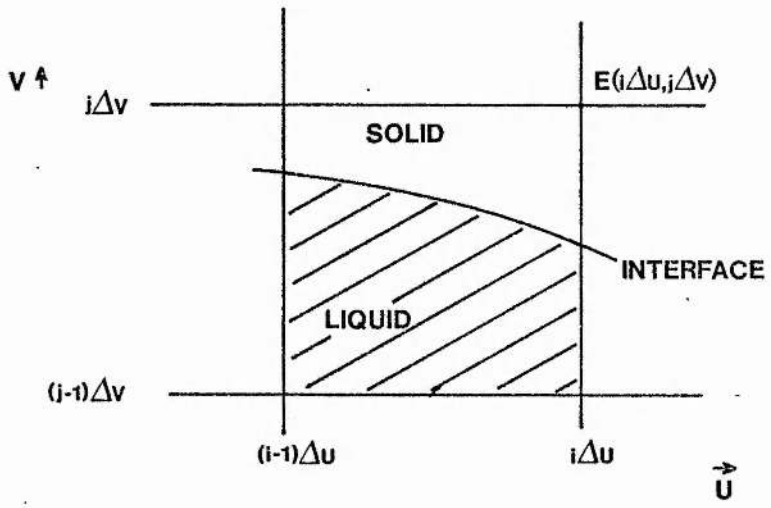
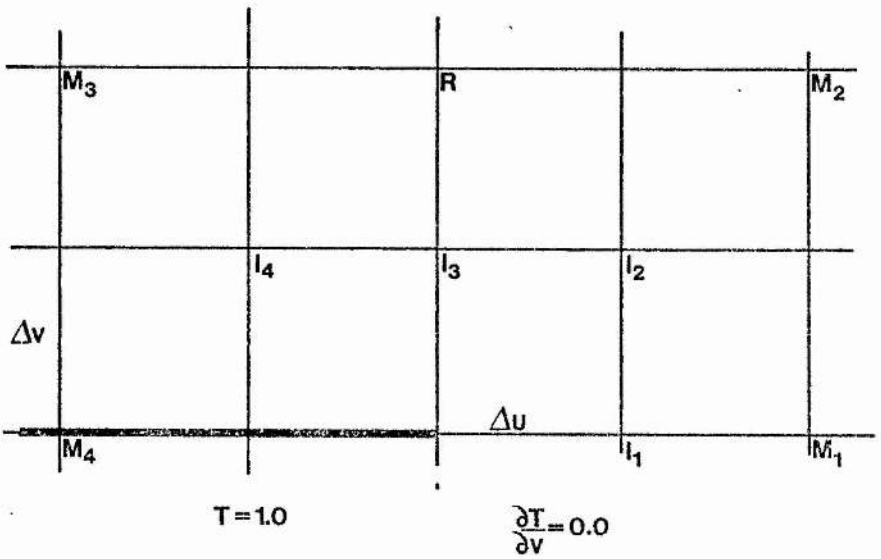


Figure 2.7

Local Difference Mesh around the Singular Point



are the number of intervals in the u and v directions respectively. To overcome the problem of an initial discontinuity (shock) in the data we use the analytic solution at some time t_0 to initialise the numerical scheme. For all examples we shall use an initial time of $t_0 = 0.5$ and a final time of $t_f = 1.0$. Using, for example, von Neumann's technique, and using the analytic solution at $t = 1.0$, it is seen that (46) is stable for $\mathcal{E} < 1/25$ which implies a time step of $\Delta t = \Delta^2/25$. To obtain the initial enthalpy distribution we can use the relations of equation (45) at each grid point within the liquid region. At points adjacent to the interface and outside the liquid region we estimate the initial enthalpies in the following way. With reference to figure 2.6 the enthalpy at the point $(i\Delta, j\Delta)$ is given as the proportion of the lower left square that is liquified. This is easily obtained by integrating the interface $v = \tanh(Y)(u_{max}^2 - u^2)^{\frac{1}{2}}$ over the segment $[(i-1)\Delta u, i\Delta u]$ and subtracting the 'excess area' below the curve. Y is the interface location of the pseudo-one-dimensional problem in the transformed (x,y,t) plane.

Before applying the numerical algorithm we should note that by (41) we would expect the difference procedure to produce a dubious accuracy in the vicinity of the singular point (u=1,v=0). Thus, in addition to providing results from the direct application of (46) (suitably modified along the boundaries $u = 0.0$ $(\partial T/\partial u = 0.0)$ and $u > 1.0$, $v = 0.0$ $(\partial T/\partial v = 0.0)$) we also consider a corrected algorithm to cater for this singular point. With reference to figure 2.7 we follow Fox and Sankar (1969) who devised a procedure for steady-state problems. For this problem we will correct the temperature at the points I_1, I_2, I_3 and I_4 which are closest to the singular point. We first apply (46) to all points in the solution region to obtain the

approximate temperature distribution at the new time. It is assumed that the series (41), truncated to three terms, holds for the rectangle $M_1M_2M_3M_4$. To obtain the coefficients a_0, a_1 and a_2 we match (41) to the newly calculated temperatures at the points M_1, M_2 and M_3. The temperatures at the points I_i, i=1,2,3,4 are then overwritten using (41). This procedure is repeated at each time step (see appendix 2.2 for the FORTRAN code).

The results at t = 1.0 (Δ=0.1, Δt=0.0004) for the analytic, difference and corrected difference solutions are displayed in table 2.1. It may be seen that the numerical solution without correction produces over estimates of the temperature, particularly so in the vicinity of the singular point (for the points I_1, I_2, I_3 and I_4 the excess is about 13.6%, 10.0%, 8.2% and 2.8% respectively). Although the corrected algorithm also produces over estimates, the disruptive effect of the singularity is considerably reduced (2.5%, 2.8%, 1.6% and 0.74% respectively). It is also clear that points some distance away from the singularity are relatively unaffected by its presence, that is, the disturbance is quite localised.

One drawback of the numerical solution is the CPU time involved. For the uncorrected algorithm 52.24 seconds of CPU time were required. For the corrected algorithm, remembering that at each time step a system of three simultaneous equations must be solved for the a_i, 56.10 seconds of CPU time were used. While a finer mesh (e.g. Δu = 0.05) produces more accurate results, the subsequent decrease of the allowable time step and increase in the number of grid points increases the CPU time (by approximately x16 for Δu = 0.05) making the scheme rather expensive computationally.

To partially overcome this problem of a large CPU time

Table 2.1

Temperature distribution for the 'test' problem at intervals of 2Δ as predicted by the enthalpy/explicit difference algorithm for $t = 1.0$ and $\varepsilon = 0.04$ ($\Delta = 0.1$, $\Delta t = 0.0004$)

065	061	050	033	010	0	0	0	0	0
076	074	064	047	025	0	0	0	0	0
077	075	066	050	029	0	0	0	0	0
148	144	130	109	079	044	004	0	0	0
160	156	143	122	093	058	022	0	0	0
162	158	146	125	097	064	025	0	0	0
246	241	225	198	161	116	066	013	0	0
257	252	237	211	176	132	083	028	0	0
260	255	240	215	180	137	087	030	0	0
361	355	336	304	259	201	136	069	004	0
371	365	347	317	273	217	153	087	024	0
374	368	351	322	279	224	160	093	028	0
495	489	469	432	377	302	215	128	047	0
503	497	478	443	390	317	232	146	065	0
506	500	482	449	398	328	244	157	076	0
649	643	624	588	525	427	304	187	086	004
655	649	631	596	536	441	321	205	106	024
657	652	635	603	547	457	338	220	118	034
819	816	804	779	724	593	397	236	115	022
822	819	807	783	731	606	413	255	135	041
823	820	810	788	742	635	440	274	148	048
						450	257	126	029
						469	277	146	045
						511	299	160	060

KEY

Analytic solution
Singularity accounted for
Singularity ignored

requirement we consider a two-dimensional application of the hopscotch algorithm. As seen in chapter I this scheme provides a stable solution for an extended range of $\mathcal{E}$ thus reducing the required number of calculations for a given mesh size. Using the same notation ($E^m_{i,j}$ and $T^m_{i,j}$) the odd-even hopscotch approximation to equation (34) is

$$E^{m+1}_{i,j} = E^m_{i,j} + \mathcal{E}\,[(u_i+1)^2+v_j^2]^{\frac{1}{2}}[(u_i-1)^2+v_j^2]^{\frac{1}{2}} \qquad - (47)$$

$$\times\,(T^m_{i+1,j} + T^m_{i-1,j} + T^m_{i,j+1} + T^m_{i,j-1} - 4T^m_{i,j})$$

($0 \leqslant i \leqslant N_u$, $0 \leqslant j \leqslant N_v$) for points i+j+m ODD and

$$E^{m+1}_{i,j} = E^m_{i,j} + \mathcal{E}\,[(u_i+1)^2+v_j^2]^{\frac{1}{2}}[(u_i-1)^2+v_j^2]^{\frac{1}{2}} \qquad - (48)$$

$$\times\,(T^{m+1}_{i+1,j} + T^{m+1}_{i-1,j} + T^{m+1}_{i,j+1} + T^{m+1}_{i,j-1} - 4T^{m+1}_{i,j})$$

($0 \leqslant i \leqslant N_u$, $0 \leqslant j \leqslant N_v$) for points i+j+m EVEN where (47) and (48) are equivalent to fully explicit and implicit approximations respectively. Suitable modifications are used on the boundaries $u = 0.0$ $(\partial T/\partial u = 0.0)$ and $u > 1.0$, $v = 0.0$ $(\partial T/\partial v = 0.0)$. We first apply (47) at all the relevant points in the solution region. To then apply (48) to the remaining points we adopt the following procedure, similar to that in chapter I. First we calculate a value $\bar{E}^{m+1}_{i,j}$ given by

$$\bar{E}^{m+1}_{i,j} = E^m_{i,j} + \mathcal{E}\,[(u_i+1)^2+v_j^2]^{\frac{1}{2}}[(u_i-1)^2+v_j^2]^{\frac{1}{2}} \qquad - (49)$$

$$\times\,(T^{m+1}_{i+1,j} + T^{m+1}_{i-1,j} + T^{m+1}_{i,j+1} + T^{m+1}_{i,j-1})$$

All the implicit temperatures on the right-hand side of (49) have been obtained by the application of (47) at the current time step. If $\bar{E}^{m+1}_{i,j} \ll 1.0$ ($\alpha = 1.0$) then it can be shown that $E^{m+1}_{i,j} \ll 1.0$ and hence, from (45), $T^{m+1}_{i,j} = 0.0$. If, however, $\bar{E}^{m+1}_{i,j} > 1.0$ then $E^{m+1}_{i,j}$ is given by

$$E^{m+1}_{i,j} = \bar{E}^{m+1}_{i,j} + \frac{4\varepsilon[(u_i+1)^2+v_j^2]^{\frac{1}{2}}[(u_i-1)^2+v_j^2]^{\frac{1}{2}}}{1.0 + 4\varepsilon[(u_i+1)^2+v_j^2]^{\frac{1}{2}}[(u_i-1)^2+v_j^2]^{\frac{1}{2}}} \qquad - (50)$$

It is easily shown that the composite effect of using (49) and (50) is equivalent to one application of (48). The temperatures at the points $I_i, i=1,2,3,4$ are now corrected in the afore-mentioned manner. With $t_0 = 0.5$, $t_f = 1.0$, $\Delta = 0.1$ and $\varepsilon = \Delta^2/25$ ($\Delta t = 0.0004$) the algorithm produces the results displayed in table 2.2. A graphical representation of the isotherms obtained for the analytic and corrected numerical solution is given in figure 2.8. The +'s represent the boundary enthalpies for each $i\Delta u$ obtained from the numerical solution. The numerical interface location lies between these points.

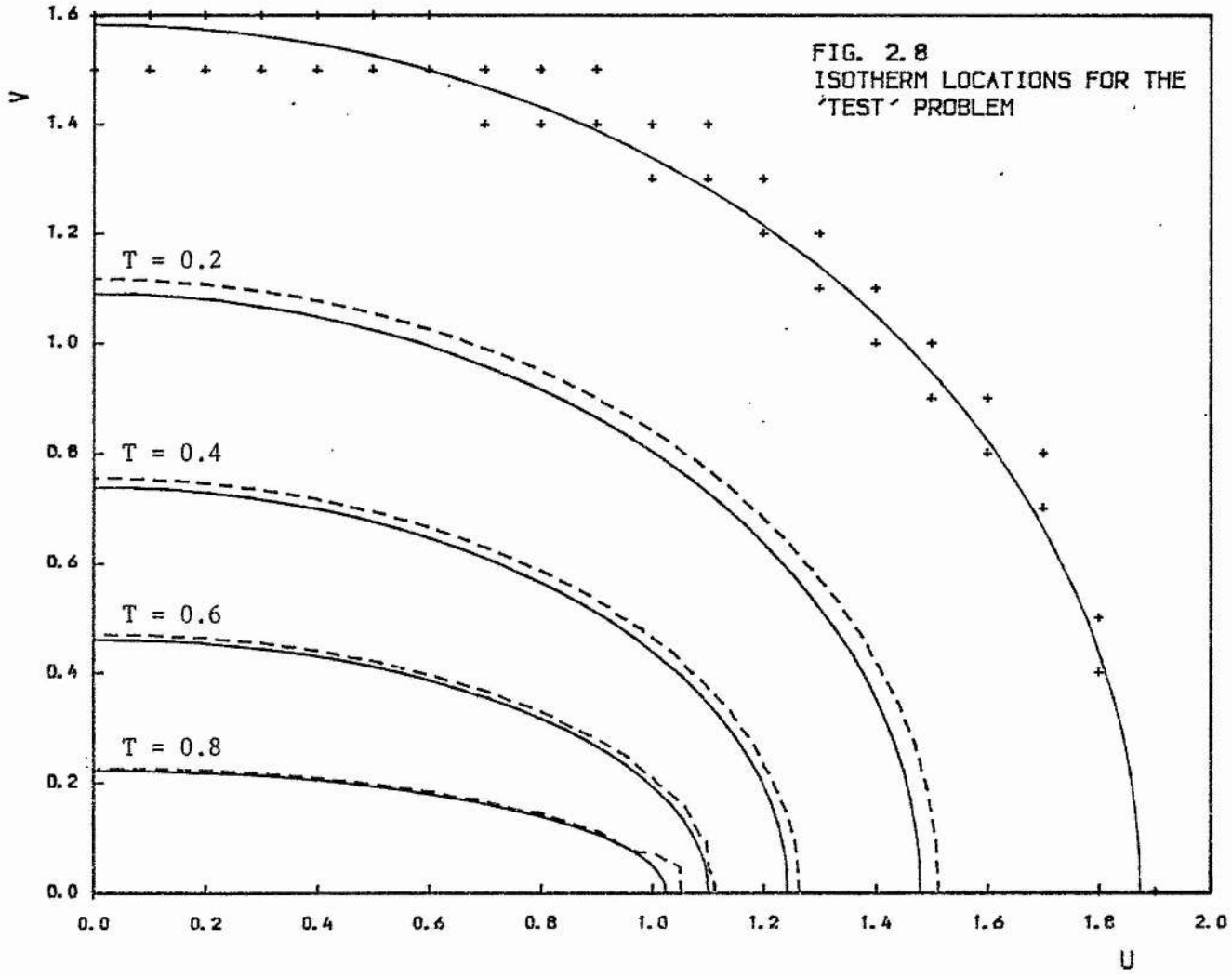
In addition to the extended parameter range of the hopscotch scheme it also lends its self to more efficient coding (see appendix 2.3 for the FORTRAN code). The CPU times for the above hopscotch implementations were 57.64 seconds and 57.83 seconds for the uncorrected and corrected algorithms respectively. We see that these values are of the order of those for the fully explicit scheme using the same parameter values. Under normal circumstances we would expect the hopscotch algorithm to be significantly slower because an extra set of calculations is required for the implicit hopscotch algorithm

Table 2.2

Temperature distribution for the 'test' problem at intervals of 2Δ as predicted by the enthalpy/hopscotch difference algorithm for $t = 1.0$ and $\varepsilon = 0.04$ ($\Delta = 0.1$, $\Delta t = 0.0004$)

065	061	050	033	010	0	0	0	0	0
077	074	066	049	027	0	0	0	0	0
078	075	067	051	030	0	0	0	0	0
148	144	131	109	079	044	004	0	0	0
161	157	144	123	094	059	022	0	0	0
163	159	147	126	098	065	025	0	0	0
246	241	225	198	161	116	066	013	0	0
258	253	238	212	177	133	083	028	0	0
261	256	241	216	181	138	087	030	0	0
361	355	336	304	259	201	136	069	004	0
372	366	348	317	273	218	154	087	025	0
374	369	352	322	280	225	161	093	028	0
495	489	469	432	377	302	215	128	047	0
504	498	478	444	390	318	233	146	066	0
507	501	482	449	399	328	244	157	076	0
649	643	624	588	525	427	304	187	086	004
655	649	632	597	537	441	321	205	106	024
657	652	635	603	547	457	338	220	118	034
819	816	804	779	724	593	397	236	115	022
822	819	808	783	731	607	414	255	136	041
824	821	810	788	742	635	440	274	148	048
						450	257	126	029
						470	277	147	045
						511	299	160	060

KEY | Analytic solution
Singularity accounted for
Singularity ignored



KEY:

\+ : These points are the last positive enthalpies given by the numerical algorithm

Broken line: Numerical isotherms

Solid line: Analytic isotherms & interface

(equation (50)) at each grid point. Most computers put a limitation on the size of array declarations (in our case we are using a three dimensional array for the u,v and t axes). To reduce the array size it is usual to declare just two 'temporal' locations to hold the (temperature) solution at two consecutive time levels ($m\Delta t$ and $(m+1)\Delta t$, say). Once the solution at the $(m+1)^{th}$ time step has been calculated for the solution region a segment of code is required to over-write the previous solution. However, because of the way in which the hopscotch algorithm moves through the spacial grid newly calculated values (of the temperature) at a node $(i\Delta u, j\Delta v)$ may immediately over-write the previous solution at that node. This pre-empts the requirement of the 'over-writing code' and also reduces the array dimensions which in turn reduces indexing time. For one-dimensional problems the resulting saving in CPU time may be small but for two-dimensions the saving is considerable because of the increased number of nodes in the solution region and because of the larger number of time steps usually required (for stability reasons). For example, this 'fast' hopscotch algorithm applied to the present problem with $\Delta = 0.1$, $\Delta t = 0.0004$ ($\mathcal{E} = 1/25$) produces a saving of approximately 18% in CPU time compared to the usual code.

The accuracy of the solution produced by the hopscotch implementation is of the same order as the explicit scheme. Around the singular point the excess estimations are 13.6%, 10.0%, 8.2% and 2.8% for the points I_1, I_2, I_3 and I_4 respectively using the uncorrected algorithm. The corrected algorithm reduces these excess values to 2.7%, 3.0%, 1.7% and 0.86% which are again comparable to those of the corrected explicit scheme.

To reduce the CPU time required the enthalpy/hopscotch algorithm

Table 2.3

Temperature distribution for the 'test' problem at intervals of 2Δ as predicted by the enthalpy/hopscotch difference algorithm for t = 1.0 and $\epsilon = 0.2$ ($\Delta = 0.1$, $\Delta t = 0.002$)

065	061	050	033	010	0	0	0	0	0
079	077	069	053	031	0	0	0	0	0
080	078	070	054	032	0	0	0	0	0
148	144	130	109	079	044	004	0	0	0
165	161	149	129	102	067	025	0	0	0
167	163	151	131	104	069	028	0	0	0
246	241	225	198	161	116	066	013	0	0
262	257	242	217	182	138	087	029	0	0
265	260	245	220	186	142	094	044	0	0
361	355	336	304	259	201	136	069	004	0
375	369	352	321	278	222	158	090	027	0
378	372	355	326	283	228	164	098	030	0
495	489	469	432	377	302	215	128	047	0
506	500	481	447	394	322	237	151	073	0
509	503	485	452	401	331	247	160	078	0
649	643	624	588	525	427	304	187	086	004
657	651	633	599	539	445	325	209	112	032
659	654	637	605	549	460	341	223	121	036
819	816	804	779	724	593	397	236	115	022
823	820	809	784	733	610	418	259	139	043
824	821	811	789	743	637	442	277	152	059
						450	257	126	029
						475	281	150	046
						512	302	165	066

KEY	Analytic solution
	Singularity accounted for
	Singularity ignored

Table 2.4

Temperature distribution for the 'test' problem at intervals of 4Δ as predicted by the enthalpy/hopscotch difference algorithm for t = 1.0 and $\varepsilon = 0.16$ ($\Delta = 0.05$, $\Delta t = 0.0004$)

065	061	050	033	010	0	0	0	0	0
074	071	060	042	019	0	0	0	0	0
074	071	060	043	019	0	0	0	0	0
148	144	130	109	079	044	004	0	0	0
157	152	139	118	089	054	013	0	0	0
157	153	140	119	091	055	017	0	0	0
246	241	225	198	161	116	066	013	0	0
254	249	233	207	171	127	077	026	0	0
255	250	234	208	173	129	079	027	0	0
361	355	336	304	259	201	136	069	004	0
368	362	344	312	268	211	147	080	014	0
369	363	345	314	270	214	150	084	019	0
495	489	469	432	377	302	215	128	047	0
501	494	475	439	385	312	226	139	058	0
502	496	476	442	389	316	231	143	063	0
649	643	624	588	525	427	304	187	086	004
653	647	629	594	532	436	315	198	098	017
654	648	631	596	537	443	322	204	103	019
819	816	804	779	724	593	397	236	115	022
821	818	806	782	728	602	408	248	128	037
822	818	807	784	733	614	420	256	134	039
						450	257	126	029
						463	269	139	042
						480	279	145	044

KEY	Analytic solution
	Singularity accounted for
	Singluarity ignored

was re-run using the same mesh size of $\Delta = 0.1$ but with $\Delta t = 0.002$. This represents a five fold increase in the stability parameter. The results are displayed in table 2.3. While the accuracy is not as good as that shown in table 2.2 the CPU time is now 11.36 seconds and 11.69 seconds for the standard and corrected algorithms respectively, and the enthalpy/hopscotch scheme still produces a valid representation of the temperature distribution.

To improve on the accuracy shown in table 2.2 we may half the mesh size and leave the time step at $\Delta t = 0.0004$ (table 2.4). The CPU time is increased by a factor of approximately x4 (to 200.0 seconds) compared with x16 for the fully explicit scheme. The relevant excess estimations at the points around the singular points are 6.8%, 5.3%, 4.2%, 1.5% and 2.3%, 2.0%, 1.4%, 0.62% for the uncorrected and corrected algorithms respectively. As one would expect these are smaller than those determined from table 2.2.

Thus, in conclusion, we have shown that the enthalpy/hopscotch algorithm, suitably modified to counter the effect of the singular point, provides a stable solution which is a fair reflection of the analytic temperature distribution of the test problem.

2.5 A Numerical Solution Of The 'Element' Problem

We now investigate the preceding numerical approach as applied to the originally posed 'element' problem of section 2.2. Utilizing the enthalpy reformulation the governing heat conduction equation (8) becomes

$$\frac{\partial E}{\partial t} = \frac{\partial^2 T}{\partial u^2} + \frac{\partial^2 T}{\partial v^2}, \quad (0 \leqslant u \leqslant 1, v > 0), (u > 1, v \geqslant 0), t > 0 \qquad - (51)$$

(for α = 1.0) where the interface location is catered for by the use of the relations (45). The explicit and implicit algorithms of the odd-even hopscotch approximation of (51) are

$$E^{m+1}_{i,j} = E^{m}_{i,j} + \varepsilon(T^{m}_{i+1,j} + T^{m}_{i,j-1} + T^{m}_{i,j+1} + T^{m}_{i-1,j} - 4T^{m}_{i,j}) \qquad - (52)$$

$(0 \leqslant i \leqslant N_u,\ 0 \leqslant j \leqslant N_v,\ m \geqslant 0)$ for points i+j+m ODD and

$$\bar{E}^{m+1}_{i,j} = E^{m}_{i,j} + \varepsilon(T^{m+1}_{i+1,j} + T^{m+1}_{i-1,j} + T^{m+1}_{i,j+1} + T^{m+1}_{i,j-1}) \qquad - (53)$$

$$E^{m+1}_{i,j} = \frac{\bar{E}^{m+1}_{i,j} + 4\varepsilon}{1.0 + 4\varepsilon} \quad \text{for } \bar{E}^{m+1}_{i,j} > 1.0 \qquad - (54)$$

$$= 0.0 \qquad \text{otherwise}$$

for points i+j+m EVEN. Equation (52) is first applied to the relevant points followed by an application of (53) and (54) at the remaining points. From (16), truncated to four terms, the corrected temperatures at the points I_i, i=1,2,3,4 (figure 2.7) are given by

$$T(r,\theta,t) \simeq 1 + a_0(t)r^{\frac{1}{2}}\sin(\theta/2) + a_1(t)r^{\frac{3}{2}}\sin(3\theta/2) \qquad - (55)$$
$$+ a_2(t)r^{\frac{5}{2}}\sin(\theta/2) + a_3(t)r^{\frac{5}{2}}\sin(3\theta/2) + O(r^{\frac{7}{2}})$$

where the four coefficients a_i are obtained by matching (55) to the numerical solution at the points M_1, M_2, M_3 and R.

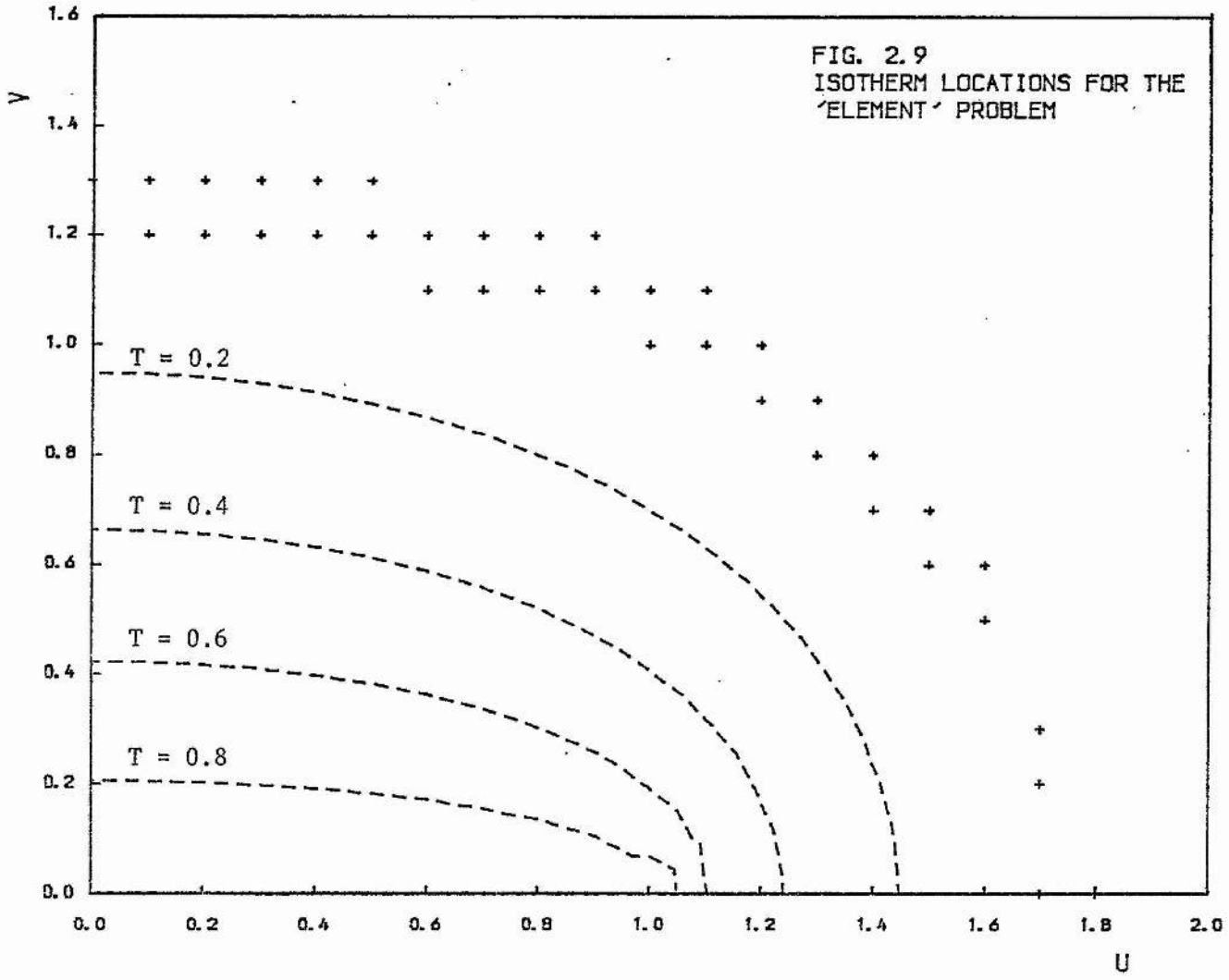
The numerical schemes are again initialised at t_0 = 0.5 using the analytic solution of the test problem which we suppose mimics the true solution. It will infact be an over estimate due to the factor

Table 2.5

Temperature distribution for the 'element' problem at intervals of 2Δ as predicted by the enthalpy/hopscotch difference algorithm for $t = 1.0$ and $\varepsilon = 0.04$ ($\Delta = 0.1$, $\Delta t = 0.0004$)

053	051	042	0	0	0	0	0	0
053	051	042	0	0	0	0	0	0
167	163	147	119	083	040	0	0	0
169	164	149	121	086	045	0	0	0
299	293	275	244	199	142	075	0	0
300	295	277	247	203	146	079	0	0
450	444	425	391	337	262	171	078	0
452	446	428	395	343	269	179	083	0
620	615	598	564	504	406	277	149	043
622	617	601	569	513	419	290	160	049
806	802	792	768	717	591	385	209	078
807	804	794	772	726	616	408	226	087
						450	236	095
						486	255	106

KEY | Singularity accounted for
Singularity ignored



KEY:

\+ : These points are the last positive enthalpies given by the numerical algorithm

Broken line: Numerical isotherms

Table 2.6

Temperature distribution for the 'element' problem at intervals of 2Δ as predicted by the enthalpy/hopscotch difference algorithm for $t = 1.0$ and $\varepsilon = 0.25$ ($\Delta = 0.1$, $\Delta t = 0.0025$)

054	052	043	0	0	0	0	0	0
054	052	043	005	0	0	0	0	0
169	164	149	120	086	045	0	0	0
170	165	150	122	090	048	0	0	0
300	294	277	246	201	143	077	0	0
302	296	279	248	205	148	080	0	0
451	445	426	392	339	264	173	079	0
453	447	429	396	344	271	180	084	0
621	616	599	565	506	408	279	152	046
622	618	601	570	513	420	291	162	050
806	803	792	769	718	594	388	213	081
807	804	794	772	727	617	409	227	088
						454	240	098
						487	257	107

KEY | Singularity accounted for
Singularity ignored

$[(u+1)^2+v^2]^{\frac{1}{2}}[(u-1)^2+v^2]^{\frac{1}{2}}$. A 'fast' coding procedure is again adopted (see appendix 2.4 for the FORTRAN code).

The results are displayed in table 2.5 and figure 2.9 for the parameter values $t_0 = 0.5$, $t_f = 1.0$, $\Delta u = \Delta v = \Delta = 0.1$, $\varepsilon = 1/25$ and $\Delta t = 0.0004$. Comparing figure 2.8 and 2.9 we see that, as expected, the temperature profile is 'flatter' for the 'element' problem and the interface is not as far advanced.

We note that the stability restriction (for the fully explicit algorithm (52)) is relaxed for this problem with the requirement that $\varepsilon \ll 1/4$. Thus, we would expect the hopscotch algorithm to produce a good approximation to the solution for considerably larger values of ε than 1/25. The results for $\Delta = 0.1$, $\Delta t = 0.0025$ ($\varepsilon = 1/4$) are displayed in table 2.6. They show a deterioration of at most three digits in the third figure compared to those in table 2.5. However, the CPU time (for both the uncorrected and corrected algorithms) is approximately 2.6 seconds as compared to 16.3 seconds for the results in table 2.5. Hence we could easily obtain solutions for much longer times with little difficulty.

For a refined mesh size (with ε constant) we would expect the numerical solution to converge to the analytic solution. For $\varepsilon = 1/4$ and $\Delta = 0.05$ (table 2.7) the uncorrected solution is seen to be lower than that of table 2.6 ($\Delta = 0.1$, $\varepsilon = 1/4$). Thus, it would seem reasonable to suppose that the lower estimates of the corrected algorithm in table 2.6 are infact closer to the true solution. Hence it is reasonable to assume that the corrected algorithm for $\Delta = 0.05$ provides a (lower) closer approximation to the solution than does the uncorrected version and that the corrected algorithm produces a converging solution as the mesh is refined (a run with $\Delta = 0.025$,

Table 2.7

Temperature distribution for the 'element' problem at intervals of 4Δ as predicted by the enthalpy/hopscotch difference algorithm for $t = 1.0$ and $\varepsilon = 0.25$ ($\Delta = 0.05$, $\Delta t = 0.000625$)

048	043	026	0	0	0	0	0	0
049	043	026	0	0	0	0	0	0
159	154	138	113	077	033	0	0	0
160	154	139	114	078	035	0	0	0
292	287	269	238	192	134	067	0	0
293	287	270	239	194	136	070	0	0
446	440	421	385	330	254	163	071	0
447	441	422	387	333	258	167	074	0
618	612	595	560	498	398	270	142	036
618	613	596	562	502	404	276	147	038
804	801	790	766	714	585	379	203	071
805	802	791	768	718	595	389	210	075
						442	229	085
						457	237	088

KEY | Singularity accounted for
Singularity ignored

Table 2.8

Temperature distribution for the 'element' problem at intervals of 2Δ as predicted by the enthalpy/hopscotch difference algorithm for $t = 1.0$ and $\varepsilon = 0.5$ ($\Delta = 0.1$, $\Delta t = 0.005$)

055	053	047	027	0	0	0	0	0
055	053	049	027	0	0	0	0	0
171	167	151	127	092	049	0	0	0
173	168	153	131	095	051	0	0	0
303	297	279	247	203	147	078	0	0
305	299	282	252	210	153	083	005	0
453	447	428	393	339	263	173	080	0
455	450	432	399	348	275	184	086	0
622	617	599	565	504	405	277	153	047
624	619	603	572	516	423	296	167	055
806	803	792	768	715	584	381	211	082
808	805	795	774	728	619	413	234	099
						440	236	098
						491	262	114

KEY — Singularity accounted for
Singularity ignored

$\varepsilon = 1/4$ follows this pattern).

Utilizing the extended parameter range of the hopscotch algorithm and taking $\Delta = 0.1$, $\Delta t = 0.005$ we see (table 2.8) that the scheme still provides a valid description of the temperature distribution in the liquid region. The CPU time is now reduced to 1.19 seconds and 1.46 seconds for the uncorrected and corrected algorithms respectively.

2.6 Conclusions

In sections 2.3 and 2.4 we described a two-dimensional test problem, with significant computational difficulties, and sucessfully applied the enthalpy method (with the hopscotch difference scheme) to obtain a valid description of the temperature distribution. Comparison with the analytic solution justifies this statement. It was also seen that with a suitable modification the effect of the 'localised' singularity on the numerical solution could be considerably reduced.

In section 2.5 we applied the same numerical algorithm to the 'element' problem of section 2.2. The similarity between the two problems is evident - two-dimensional phase change outside a heated ribbon with a singular point on the boundary. In the absence of any analytic solution, and bearing in mind the results of section 2.4, we see that the enthalpy/hopscotch approach provides a fast and converging solution of the problem. Again, the effect of the singularity is significantly reduced by using the local series modification.

Thus, in conclusion, we may have confidence in the numerical

algorithm desribed when tackling two-dimensional Stefan problem, including situations with points of difficulty in the solution region.

CHAPTER III

PHASE CHANGE IN THE REGION OUTSIDE AN ELLIPTICAL CYLINDER

3.1 Introduction

In the previous chapter the use of a conformal mapping was investigated with respect to obtaining a numerical solution for the problem of a finite heating element. In particular, the transformation $w = \sin(z)$ was used (where $w,z \in \mathbb{C}$, $w = u + iv$, $z = x + iy$ and $x,y,u,v \in \mathbb{R}$). The resulting heat diffusion equation closely resembles Mathieu's equation in elliptical co-ordinates. It is clear that an ellipse in the original plane is mapped onto a straight line in the transformed plane. This suggests the idea of solving the heat conduction problem in the region outside an infinite elliptical cylinder. The corresponding pure conduction problem for the region inside an elliptical cylinder has been solved by McLachlan (1945). The validity of his solution is justified by showing that, as the eccentricity of the bounding ellipse tends to zero, it degenerates to the Bessel function solution describing pure conduction inside a circular cylinder.

In relation to the 'element' problem it is easily shown that as the eccentricity of an ellipse tends to unity so the ellipse tends to an element of length 2h (the interfocal distance) lying along the major axis of the ellipse.

In this chapter a solution in terms of Mathieu functions is found for the case of pure conduction outside an elliptical surface. The validity of the solution is partially justified by showing a

degeneration to the Bessel function solution, for conduction outside a circular surface, as the eccentricity of the ellipse approaches zero. A corresponding solution for the heating element is then deduced by allowing the eccentricity to approach unity.

With regard to the phase-change aspect of this problem we make use of a concept first used by Lightfoot (1930). His idea was to represent the phase-change boundary as a moving source of heat. The solution to the full phase-change problem then consists of two parts. The first is the corresponding pure heat conduction problem. To this solution is added the solution of the problem of a moving line-source of heat in a medium initially at zero temperature and with zero boundary conditions. This technique has been sucessfully used by Rathjen and Jiji (1971) who consider the problem of phase-change in an infinite two-dimensional corner. Section 3.3 deals with this 'moving heat-source' problem.

3.2 The Pure Conduction Problem

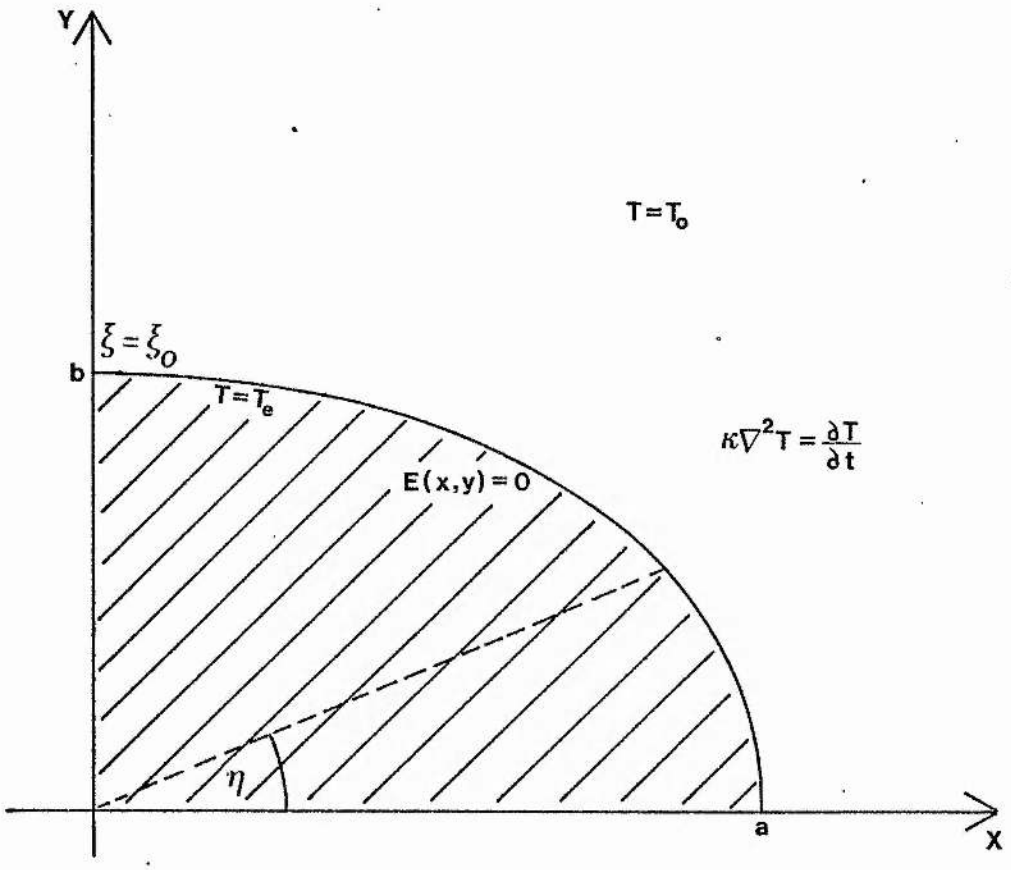
We consider an infinite elliptical cylinder whose axis is fixed to lie along the z-axis in the usual rectangular co-ordinate system (x,y,z) and whose surface is given by the equation

$$E(x,y) = \frac{x^2}{a^2} + \frac{y^2}{b^2} - 1 = 0$$

where a and b are, respectively, the semi-major and semi-minor axes of the ellipse. By the symmetry of the problem we need only consider the positive quarter plane, (figure 3.1). We consider the region surrounding the ellipse to have an initial temperature T_o. At time

Figure 3.1

Heat Conduction Outside an Elliptic Cylinder



Cartesian co-ordinates - $E(x,y) = \frac{x^2}{a^2} + \frac{y^2}{b^2} - 1 = 0$

Elliptic co-ordinates - $\xi = \xi_o$

$t = 0$ the surface of the cylinder is subject to a constant temperature T_e which is maintained for all subsequent time. The ensuing temperature distribution in the surrounding region is governed by the two-dimensional time-dependent heat conduction equation which describes a set of isotherms propagating outwards from the surface of the cylinder.

Mathematically the temperature distribution in the region $E(x,y) > 0$ (for $t > 0$) is described by

$$\frac{\partial T}{\partial t} = \kappa \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right) , \quad E(x,y) > 0, \; t > 0$$

subject to the initial and boundary conditions

$$T(x,y,0) = T_o , \quad E(x,y) > 0, \; t = 0$$
$$T(x,y,t) = T_e , \quad E(x,y) = 0, \; t > 0$$

From asymptotic considerations we also have

$$T(x,y,t) \rightarrow T_o , \quad |x| \text{ or } |y| \rightarrow \infty$$

for any time t. That is, at a large distance from the cylinder the temperature approaches its initial value. For convenience the temperature may be scaled by making the change of variable

$$U = \frac{(T - T_o)}{(T_e - T_o)}$$

This yields

$$\frac{\partial U}{\partial t} = K \left(\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} \right) , \quad E(x,y) > 0, \; t > 0$$

subject to

$$U(x,y,0) = 0 , \quad E(x,y) > 0, \; t = 0$$

$$U(x,y,t) = 1 , \quad E(x,y) = 0, \; t > 0$$

$$U(x,y,t) \to 0 , \quad |x| \text{ or } |y| \to \infty , \; t > 0$$

We now transform the problem into elliptical cylindrical co-ordinates (ξ,η) using the change of variables

$$x = h \cosh\xi \cos\eta$$

$$y = h \sinh\xi \sin\eta$$

where $2h$ is the interfocal distance of the resulting set of confocal ellipses and hyperbolae describing the new co-ordinate system. By simple manipulation we obtain

$$\frac{h^2}{2K} \left(\cosh 2\xi - \cos 2\eta \right) \frac{\partial U}{\partial t} = \frac{\partial^2 U}{\partial \xi^2} + \frac{\partial^2 U}{\partial \eta^2} \qquad - (1)$$

$$\xi > \xi_o, \; 0 \leqslant \eta \leqslant 2\pi, \; t > 0$$

subject to

$$U(\xi,\eta,0) = 0 , \quad \xi > \xi_o , \; 0 \leqslant \eta \leqslant 2\pi, \; t = 0 \qquad - (1a)$$

$$U(\xi_o,\eta,t) = 1 , \quad \xi = \xi_o , \; 0 \leqslant \eta \leqslant 2\pi, \; t > 0 \qquad - (1b)$$

$$U(\xi,\eta,t) \to 0 , \quad \xi \to \infty , \; 0 \leqslant \eta \leqslant 2\pi, \; t > 0 \qquad - (1c)$$

where ξ_o is the 'radial' co-ordinate of the bounding ellipse.

The inclination now is to follow McLachlan (1945) and set

$$U(\xi,\eta,t) = f(\xi,\eta)e^{-\kappa\nu^2 t}$$

where f is a function of ξ and η only. In (1) this substitution leads to a separable equation in $f(\xi,\eta)$ from which the cannonical forms of Mathieu's equation and modified equation may be deduced. The 'angular' η-solution yields an even, single-valued function of period π (by the symmetry of $U(\xi,\eta,t)$ about the cylinder). By (1c) the ξ-solution required is a modified radial (or second) type of Mathieu function, that is, one which is <u>not periodic</u> in ξ. The formal solution is then

$$f(\xi,\eta) = \sum_{m=0}^{\infty} C_{2m}\mathcal{X}_{2m}(\xi)\,\Psi_{2m}(\eta)$$

where $\mathcal{X}_{2m}(\xi)$ and $\Psi_{2m}(\eta)$ represent the 'radial' and 'angular' solutions respectively. For conduction inside an elliptical cylinder $\mathcal{X}_{2m}(\xi)$ is a modified Mathieu function of the first kind and the constants C_{2m} may be determined by utilizing an orthogonality theorem involving these Mathieu functions [see McLachlan (1947), p.176].

This method does not appear to be applicable for the current problem since there is no similar orthogonality theorem for the modified radial Mathieu functions. We thus require to develop the solution in another way.

Since $t \in (0,\infty)$ and noting the form of time dependence ($e^{-\kappa\nu^2 t}$) chosen by McLachlan (1945) it would appear reasonable to explore the

possibilities of using Laplace transforms. This approach is furthered since $U(t = 0) = 0$ and the transformation of the term $\frac{\partial U}{\partial t}$ will produce the zero term $[U]_{t=0}$. Utilizing the definitions of Laplace transforms [see Doetsch (1961), p.31 and p.37]

$$L\ [\ U(\xi,\eta,t)\ ] = f(\xi,\eta;s) = \int_0^\infty e^{-st}\ U(\xi,\eta,t)\ dt$$

and

$$L\ [\ \frac{\partial U}{\partial t}(\xi,\eta,t)\ ] = s\ f(\xi,\eta;s) - U(\xi,\eta,0+)$$

we obtain from (1) and (1a)

$$\frac{h^2 s}{2K}\ (\ \cosh 2\xi - \cos 2\eta\ )\ f = \frac{\partial^2 f}{\partial \xi^2} + \frac{\partial^2 f}{\partial \eta^2} \qquad - (2)$$

$$\xi > \xi_o,\ 0 \leqslant \eta \leqslant 2\pi$$

The remaining conditions (1b) and (1c) transform to

$$f(\xi_o,\eta;s) = \frac{1}{s}\ ,\quad \xi = \xi_o,\ 0 \leqslant \eta \leqslant 2\pi \qquad - (2a)$$

$$f(\xi,\eta;s) \to 0\ ,\quad \xi \to \infty\ ,\ 0 \leqslant \eta \leqslant 2\pi \qquad - (2b)$$

Equation (2) is a separable equation for $f(\xi,\eta;s)$ in ξ and η. Setting $f(\xi,\eta;s) = \chi(\xi;s)\ \Psi(\eta;s)$ yields the pair of equations

$$\frac{d^2\chi}{d\xi^2} - (\ a - 2k^2\ \cosh 2\xi\ )\chi = 0\ ,\quad \xi > \xi_o,\ 0 \leqslant \eta \leqslant 2\pi \qquad - (2c)$$

$$\frac{d^2\Psi}{d\eta^2} + (\ a + 2k^2\ \cos 2\eta\ )\Psi = 0\ ,\quad \xi > \xi_o,\ 0 \leqslant \eta \leqslant 2\pi \qquad - (2d)$$

where 'a' is a separation constant and $2k^2 = \frac{h^2 s}{2K}$. This pair of equations correspond to the cannonical form of Mathieu's modified equation and Mathieu's equation with $2k^2$ being replaced by $-2k^2$ [McLachlan (1947), p.21]. Considering the symmetric nature of $U(\xi,\eta,t)$ a suitable solution of (2d) is

$$\Psi(\eta;s) \propto ce_{2m}(\eta,-k^2)$$

where ce_{2m} is an even Mathieu's function of the first kind with period π. For the 'radial' solution, remembering the asymptotic behaviour (2b), a relevant Mathieu function is [McLachlan (1947), p.221]

$$\chi(\xi;s) \propto Fek_{2m}(\xi,-k^2)$$

Thus the formal solution is

$$f(\xi,\eta;s) = \sum_{m=0}^{\infty} C_{2m}\, Fek_{2m}(\xi,-k^2)\, ce_{2m}(\eta,-k^2)$$

where $m = 0,1,2,.....$ correspond to the legitimate values $a_0,a_1,a_2,.....$ that the separation constant may take. These 'characteristic numbers' a_m are polynomials in k.

To evaluate the constants C_{2m} we utilize the boundary condition (2a) and obtain

$$f(\xi_o,\eta;s) = \frac{1}{s} = \sum_{m=0}^{\infty} C_{2m}\, Fek_{2m}(\xi_o,-k^2)\, ce_{2m}(\eta,-k^2) \qquad - (3)$$

the right-hand-side being a function of η only. The function

$ce_{2m}(\eta,-k^2)$ is subject to the orthogonality conditions [McLachlan (1947), p.23]

$$\int_0^{2\pi} ce_{2m}(z,q)\, ce_{2p}(z,q)\, dz = 0 , \quad m \neq p$$

and

$$\int_0^{2\pi} ce_{2m}^2(z,q)\, dz = \pi \left(2\left[A_0^{(2m)} \right]^2 + \sum_{r=1}^{\infty} \left[A_{2r}^{(2m)} \right]^2 \right)$$

which hold for all q. By convention the Mathieu functions are normalised as follows

$$\int_0^{2\pi} ce_{2m}^2(z,q)\, dz = \pi$$

From (3) we construct

$$\int_0^{2\pi} \frac{ce_{2m}(\eta,-k^2)}{s}\, d\eta = C_{2m}\, Fek_{2m}(\xi_o,-k^2) \int_0^{2\pi} ce_{2m}^2(\eta,-k^2)\, d\eta \qquad - (3a)$$

Using the definition [McLachlan (1947), p.21]

$$ce_{2m}(z,-q) = (-1)^m \sum_{r=0}^{\infty} (-1)^r A_{2r}^{(2m)} \cos 2rz$$

(where the $A_{2r}^{(2m)}$ are functions of q) we have

$$\int_0^{2\pi} ce_{2m}(\eta,-k^2)\, d\eta = (-1)^m A_0^{(2m)} \int_0^{2\pi} d\eta + (-1)^m \sum_{r=1}^{\infty} (-1)^r A_{2r}^{(2m)} \int_0^{2\pi} \cos 2r\eta\, d\eta$$

$$= 2\pi (-1)^m A_0^{(2m)}$$

Hence (3a) becomes

$$\frac{2\pi(-1)^m A_0^{(2m)}}{s} = C_{2m}\, Fek_{2m}(\xi_o,-k^2)\, \pi\Big(2[\, A_0^{(2m)}\,]^2 + \sum_{r=1}^{\infty} [\, A_{2r}^{(2m)}\,]^2 \Big)$$

and using [McLachlan (1947), p.24]

$$2[\, A_0^{(2m)}\,]^2 + \sum_{r=1}^{\infty} [\, A_{2r}^{(2m)}\,]^2 = 1$$

we have

$$C_{2m} = \frac{2\,(-1)^m A_0^{(2m)}}{s\, Fek_{2m}(\xi_o,-k^2)}$$

Thus, the transformed solution is given by

$$f(\xi,\eta;s) = \frac{2}{s}\sum_{m=0}^{\infty} (-1)^m A_0^{(2m)} \frac{Fek_{2m}(\xi,-k^2)}{Fek_{2m}(\xi_o,-k^2)}\, ce_{2m}(\eta,-k^2)$$

with $\xi \geqslant \xi_o$ and $0 \leqslant \eta \leqslant 2\pi$. We now require to use the Mellin inversion theorem to obtain the solution in (ξ,η,t) space. From Doetsch (1961) p.32 we have

$$U(\xi,\eta,t) = L^{-1}[\, f(\xi,\eta;s)\,] = \frac{1}{2\pi i}\int_{x-i\infty}^{x+i\infty} e^{ts} f(\xi,\eta;s)\, ds$$

where $s \in \mathbb{C}$ and $i^2 = -1$. Application to this problem yields

$$U(\xi,\eta,t) = \frac{1}{\pi i}\int_{a-i\infty}^{a+i\infty} \sum_{m=0}^{\infty} (-1)^m A_0^{(2m)} \frac{Fek_{2m}(\xi,-k^2)}{Fek_{2m}(\xi_o,-k^2)}\, ce_{2m}(\eta,-k^2)\, \frac{e^{ts}}{s}\, ds$$

The integration is not so straight forward as might first seem since $k = k(s)$ from $2k^2 = \frac{h^2 s}{2K}$. That is, $A_0^{(2m)}$, Fek_{2m} and ce_{2m} are functions of s.

The integrand has a singular (branch) point at $s = 0$ and, to ensure that the integrand is analytic, we take the branch line as the negative real axis. We construct a suitable contour (figure 3.2) and form a line integral. Since the integrand is analytic inside the closed contour Γ then by Cauchy's Theorem

$$\frac{1}{\pi i} \oint_{\Gamma} e^{ts} f(\xi,\Lambda;s)\, ds = 0.$$

That is, omitting the integrand

$$\frac{1}{\pi i} \left(\int_{AB} + \int_{BC} + \int_{CD} + \int_{DE} + \int_{EF} + \int_{FA} \right) = 0 \qquad - (4)$$

The integrals are suitably evaluated in the limiting case of $R \to \infty$ and $\varepsilon \to 0$.

On BC and FA we set $s = Re^{i\theta}$ and in the limit as $R \to \infty$ so the integrals over these line segments tend to zero (see appendix 3.1(i)). Also,

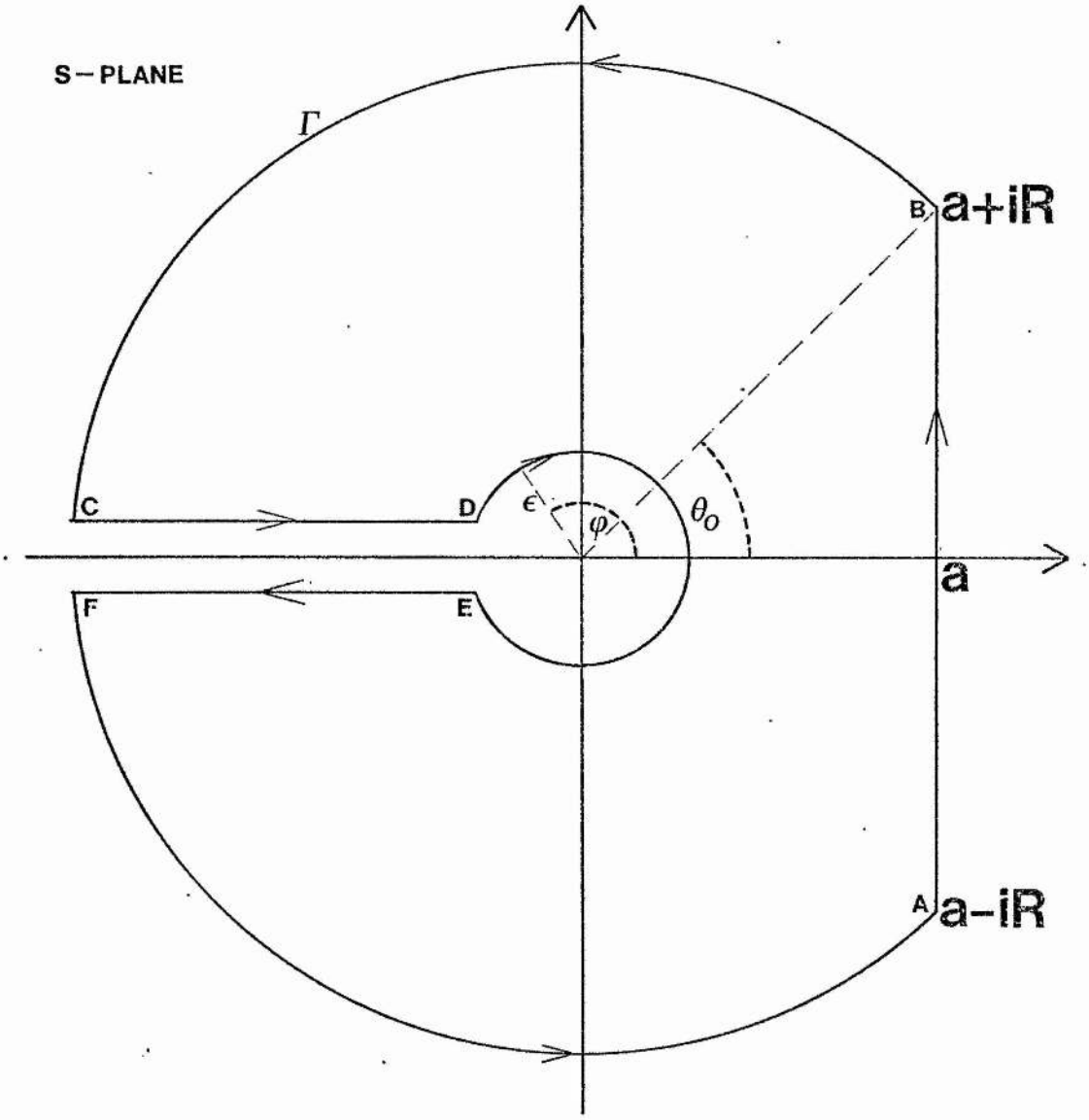
$$\lim_{R\to\infty} \int_{AB} e^{ts} f(\xi,\Lambda;s)\, ds = \pi i\, U(\xi,\Lambda,t) \qquad - (5)$$

On the line segment CD we let $s = \rho e^{i\pi}$ and the corresponding integral over CD becomes

$$\int_{\infty}^{0} e^{\rho e^{i\pi} t} f(\xi,\Lambda;\rho e^{i\pi})\, d\rho$$

Figure 3.2

Contour for the Mellin Inversion



which evaluates to (appendix 3.1(ii))

$$-\sum_{m=0}^{\infty} \int_{o}^{\infty} \frac{e^{-\alpha\omega t} A_0^{(2m)} F_{2m}(\xi,\omega) ce_{2m}(\eta,\omega)}{[Ce_{2m}^2(\xi_o,\omega) + Fey_{2m}^2(\xi_o,\omega)]} \frac{d\omega}{\omega} \qquad - (6)$$

$$-i \sum_{m=0}^{\infty} \int_{o}^{\infty} \frac{e^{-\alpha\omega t} A_0^{(2m)} G_{2m}(\xi,\omega) ce_{2m}(\eta,\omega)}{[Ce_{2m}^2(\xi_o,\omega) + Fey_{2m}^2(\xi_o,\omega)]} \frac{d\omega}{\omega}$$

where, for convenience, $\alpha = 4\kappa/h^2$ and

$$F_{2m}(\xi,\omega) = Ce_{2m}(\xi,\omega)\, Ce_{2m}(\xi_o,\omega) + Fey_{2m}(\xi,\omega)\, Fey_{2m}(\xi_o,\omega) \qquad - (6a)$$

$$G_{2m}(\xi,\omega) = Ce_{2m}(\xi,\omega)\, Fey_{2m}(\xi_o,\omega) - Fey_{2m}(\xi,\omega)\, Ce_{2m}(\xi_o,\omega) \qquad - (6b)$$

On the line segment EF we let $s = \rho e^{-i\pi}$ and the integral over EF becomes

$$\int_{o}^{\infty} e^{\rho e^{-i\pi} t} f(\xi,\eta;\rho e^{-i\pi})\, d\rho$$

which evaluates to the negative complex conjugate of (6), (appendix 3.1(ii)). Thus, the contribution from the line segments CD and EF is

$$- 2i \sum_{m=0}^{\infty} \int_{o}^{\infty} \frac{e^{-\kappa\omega t} A_0^{(2m)} G_{2m}(\xi,\omega) ce_{2m}(\eta,\omega)}{[Ce_{2m}^2(\xi_o,\omega) + Fey_{2m}^2(\xi_o,\omega)]} \frac{d\omega}{\omega} \qquad - (7)$$

On the small circle DE we let $s = \varepsilon e^{i\phi}$ and integrate between $\phi = \pi$ and $\phi = -\pi$. In the limit as the radius ε tends to zero we have (appendix 3.1(iii))

$$\lim_{\varepsilon \to 0} \int_{DE} e^{ts} f(\xi,\eta;s)\, ds = -\pi i \qquad - (8)$$

From (4),(5),(7) and (8) we have

$$U(\xi,\eta,t) = \frac{1}{\pi i} \int_{AB} = -\frac{1}{\pi i} \left(\int_{DE} + \int_{CD} + \int_{EF} \right)$$

Thus, the normalised temperature distribution outside the elliptical cylinder for pure heat conduction is given by

$$U(\xi,\eta,t)\ 1 + \frac{2}{\pi} \sum_{m=0}^{\infty} \int_{o}^{\infty} \frac{e^{-\kappa\omega t} A_0^{(2m)} G_{2m}(\xi,\omega) ce_{2m}(\eta,\omega)\, d\omega}{[Ce_{2m}^2(\xi_o,\omega) + Fey_{2m}^2(\xi_o,\omega)]\ \omega}$$

$$\xi_o \leqslant \xi < \infty\ ,\ 0 \leqslant \eta \leqslant 2\pi,\ t \geqslant 0 \qquad - (9)$$

3.2.1 A Corresponding Circular Cylinder Problem

It is the purpose of this section to validate the above series solution (9) by examining the solution as the eccentricity of the bounding ellipse tends to zero. The major and minor extremities of an

ellipse are given by $r = \pm h \cosh\xi$ and $s = \pm h \sinh\xi$ respectively. For an eccentricity e, $h = re$ and $\cosh\xi = \frac{r}{h} = \frac{1}{e}$. As $e \to 0$ the ellipse tends to a circle. For a constant value of r, $h \to 0$. For fixed r this implies that $\cosh\xi \to \infty$ and hence $\xi \to \infty$. For large ξ, $h \cosh\xi \simeq \frac{he^{\xi}}{2} \simeq h \sinh\xi$. That is, the semi-major and semi-minor lengths r and s tend to the same value $\frac{he^{\xi}}{2}$ in the limiting process $h \to 0$ and $\xi \to \infty$. In particular, $h \cosh\xi_0 \to r_0$ and $h \sinh\xi_0 \to r_0$.

With regards to the solution (9) we note that $\omega = \frac{h^2 p}{4K}$ thus $\omega \to 0$ as $h \to 0$ and [McLachlan (1947), § 2.21]

$$\lim_{\omega \to 0} A_0^{(2m)} = 0 , \quad m \neq 0$$

$$= \frac{1}{\sqrt{2}} , \quad m = 0 \qquad - (10)$$

Thus, in the solution (9), only the term corresponding to m = 0 is non-zero as the ellipse approaches a circle.

From McLachlan (1947) §2, p.367 $Ce_0(\xi,\omega)$ is a solution of

$$y'' - (a - 2k^2 \cosh 2\xi) y = 0$$

which, as the ellipse of semi-major axis r tends to a circle of radius r, degenerates to

$$y'' + k^2 e^{2\xi} y = 0 \qquad - (11)$$

where $k = \omega^{\frac{1}{2}}$. Setting $2k = k_1 h$ and noting that in the limit $\frac{he^{\xi}}{2} \to r$, (11) becomes the Bessel equation

$$\frac{d^2y}{dr^2} + \frac{1}{r}\frac{dy}{dr} + k_1^2 y = 0$$

with solution $p_0' J_0(k_1 r)$. Using the definition of ω (appendix 3.1) it is easily shown that $k_1 = (\rho / K)^{\frac{1}{2}}$ and hence as the ellipse tends to a circle

$$Ce_0(\xi,\omega) \rightarrow p_0' J_0[(\rho/K)^{\frac{1}{2}} r] \qquad - (12)$$

where p_0' is a constant multiplier that is independent of r. Similarly

$$Ce_0(\xi_o,\omega) \rightarrow p_0' J_0[(\rho/K)^{\frac{1}{2}} r_o] \qquad - (13)$$

$$Fey_0(\xi,\omega) \rightarrow p_0' Y_0[(\rho/K)^{\frac{1}{2}} r] \qquad - (14)$$

$$Fey_0(\xi_o,\omega) \rightarrow p_0' Y_0[(\rho/K)^{\frac{1}{2}} r_o] \qquad - (15)$$

By McLachlan (1947) § 1, p.367 $ce_0(\eta,\omega) \rightarrow 2^{-\frac{1}{2}}$ as $\omega \rightarrow 0$. Hence, noting that $\frac{d\omega}{\omega} = \frac{d\rho}{\rho}$, the series solution (9) degenerates to

$$U(r,t) = 1 + \frac{1}{\pi} \int_0^\infty \frac{e^{-t} [H_0(r,r_o) - H_0(r_o,r)]}{J_0^2[(\rho/K)^{\frac{1}{2}} r_o] + Y_0^2[(\rho/K)^{\frac{1}{2}} r_o]} \frac{d\rho}{\rho}$$

$$r \geqslant r_o \text{ and } t \geqslant 0 \qquad - (16)$$

where

$$H_0(r,r_o) = J_0[\ (\rho/K)^{\frac{1}{2}} r\ ]\ Y_0[\ (\rho/K)^{\frac{1}{2}} r_o\ ]$$

Now let $\bar{\omega} = (\rho/K)^{\frac{1}{2}}$ then $\frac{d\rho}{\rho} = 2\ \frac{d\bar{\omega}}{\bar{\omega}}$ and (16) becomes

$$U(r,t) = 1 + \frac{2}{\pi} \int_0^{\infty} \frac{e^{-K\bar{\omega}^2 t}\ [\ J_0(\bar{\omega}r)\ Y_0(\bar{\omega}r_o) - Y_0(\bar{\omega}r)\ J_0(\bar{\omega}r_o)\ ]}{[\ J_0^2(\bar{\omega}r_o) + Y_0^2(\bar{\omega}r_o)\ ]}\ \frac{d\bar{\omega}}{\bar{\omega}} \qquad - (17)$$

$$r \geqslant r_o \text{ and } t \geqslant 0$$

which is the normalised solution for pure heat conduction in the region outside a circular cylinder of radius r_o [see Carslaw and Jaeger (1959) 13.5, p.335].

By considering the limit as the eccentricity of the elliptic problem tends to zero we have shown that the series solution (9) degenerates to the corresponding circular solution. This provides some justification that (9) is a valid solution for the elliptic pure conduction problem.

3.2.2 A Corresponding 'Flat Plate' Problem

This section considers the series solution (9) as applied to the case of a finite element lying along the major axis of the ellipse. As the eccentricity of the bounding ellipse approaches unity so this surface tends to a 'ribbon' of length 2h. If r_o is the major extremity of the ellipse then using $h = r_o e$, $r_o \to h$ as $e \to 1$. Also $\cosh\delta_o \to 1$ which implies $\delta_o \to 0$. Thus, from (9), the corresponding solution for pure heat conduction in the region surrounding a finite heating element is

$$U(\xi,\eta,t) = 1 + \frac{2}{\pi}\sum_{m=0}^{\infty}\int_{0}^{\infty}\frac{e^{-\alpha\omega t}\,A_0^{(2m)}\,G_{2m}(\xi,\omega)\,ce_{2m}(\eta,\omega)}{[\,Ce_{2m}^2(0,\omega) + Fey_{2m}^2(0,\omega)\,]}\,\frac{d\omega}{\omega}$$

$$0 \leqslant \xi < \infty\,,\quad 0 \leqslant \eta \leqslant 2\pi,\quad t \geqslant 0 \qquad -(18)$$

where G_{2m} is defined by equation (6b). $Fey_{2m}(0,\omega)$ is unbounded at $\omega = 0$. However, the integrand of (18) is analytic for $0 \leqslant \omega < \infty$ and is zero at the lower limit of integration.

3.3 The Solution For A Moving Source

To complete the solution of equations (1) to (1c) for the phase change problem we now require a solution of the same equations in which the latent heat of fusion is represented as a moving heat source located on the solid/liquid interface. This technique was first used by Lightfoot (1930) for the one-dimensional problem in connection with the solidification of molten steel. It has since been used by Rathjen and Jiji (1971) who considered the problem of heat conduction with phase change in a corner.

This approach first requires the Green's function solution for a unit instantaneous point source (at time $t = t'$) in the solution region subject to 'zero' boundary and initial conditions. Using this function and assuming the freezing front to be a line source consisting of a set of differential heat sources the required moving source solution may be found. Finding a Green's function can often be a difficult problem in its own right and here we use results quoted

for the corresponding cylindrical problem to aid us. Once the solution has been determined the complete temperature distribution is given as the sum of the pure conduction (section 3.2) and moving source solutions, $U(\xi,\eta,t) + V(\xi,\eta,t)$. To determine the freezing front location we substitute the fusion temperature T_f (assumed to be unique) into the solution to obtain an integro-differential equation of which the solution provides the interface location.

With regard to the temperature $V(\xi,\eta,t)$, suitably normalised as is $U(\xi,\eta,t)$, we know it must satisfy the conduction equation

$$\frac{h^2}{2K}(\cosh 2\xi - \cos 2\eta)\frac{\partial V}{\partial t} = \frac{\partial^2 V}{\partial \xi^2} + \frac{\partial^2 V}{\partial \eta^2} \qquad - (19)$$

$$\xi > \xi_o,\ 0 \leqslant \eta \leqslant 2\pi,\ t > 0$$

From the fact that the overall temperature distribution is given by the sum $U(\xi,\eta,t) + V(\xi,\eta,t)$ and using the conditions (1a) to (1c) we see that $V(\xi,\eta,t)$ is subject to the conditions

$$V(\xi,\eta,0) = 0,\ \xi > \xi_o,\ 0 \leqslant \eta \leqslant 2\pi,\ t = 0 \qquad - (19a)$$

$$V(\xi_o,\eta,t) = 0,\ \xi = \xi_o,\ 0 \leqslant \eta \leqslant 2\pi,\ t > 0 \qquad - (19b)$$

$$V(\xi,\eta,t) \to 0,\ \xi \to \infty,\ 0 \leqslant \eta \leqslant 2\pi,\ t > 0 \qquad - (19c)$$

At the interface curve $\xi = S(\eta,t)$ we also have

$$U(S(\eta,t),\eta,t) + V(S(\eta,t),\eta,t) = \frac{(T_f - T_o)}{(T_e - T_o)} \qquad - (19d)$$

and the interface condition

$$\frac{h^2}{2K}(\cosh 2\xi - \cos 2\eta)\frac{\partial S}{\partial t} = k\phi\left(\frac{\partial V}{\partial \xi}\bigg|_{S^+} - \frac{\partial V}{\partial \xi}\bigg|_{S^-}\right)\left(1 + \left(\frac{\partial S}{\partial \eta}\right)^2\right) \qquad - (19e)$$

where $\phi = \frac{c}{L}(T_e - T_o)$ and the conductivity k is assumed constant. S^+ and S^- imply evaluation at the interface inside the solid and liquid regions respectively.

3.3.1 The Green's Function For A Unit Point Source

In obtaining the Green's function solution, $W(\xi,\eta,t)$, for a unit point source at (ξ',η',t') we must bear in mind the conditions to be satisfied. On the bounding ellipse ($\xi = \xi_o$) and for $\xi \to \infty$ we require $W(\xi,\eta,t)$ to be zero for all time. Also $\lim_{t\to t'} W(\xi,\eta,t) = 0$ except at the point (ξ',η'). Mathematically we are looking for a solution $W(\xi,\eta,t)$ of the conduction equation

$$\frac{\partial^2 W}{\partial \xi^2} + \frac{\partial^2 W}{\partial \eta^2} = \frac{h^2}{2K}(\cosh 2\xi - \cos 2\eta)\frac{\partial W}{\partial t} \qquad - (20)$$

$$\xi > \xi_o,\ 0 \leq \eta \leq 2\pi,\ t > 0$$

subject to

$$W(\xi_o,\eta,t) = 0\ ,\quad \xi = \xi_o,\ 0 \leq \eta \leq 2\pi,\ t > 0 \qquad - (21a)$$

$$\lim_{\xi\to\infty} W(\xi,\eta,t) = 0\ ,\quad 0 \leq \eta \leq 2\pi,\ t > 0 \qquad - (21b)$$

$$\lim_{t\to t'} W(\xi,\eta,t) = 0\ ,\quad \xi \neq \xi',\ \eta \neq \eta' \qquad - (21c)$$

where (21c) takes the form

$$\frac{e^{-R^2/4K(t-t')}}{4\pi(t-t')K}$$

at the point (ξ',η'). R is given by $R^2 = (x-x')^2 + (y-y')^2$ with

$x = h \cosh\xi \cos\eta$ $(x' = h \cosh\xi' \cos\eta')$ and $y = h \sinh\xi \sin\eta$ $(y' = h \sinh\xi' \sin\eta')$ in elliptic co-ordinates. To obtain the Green's function for this problem we utilize the corresponding cylindrical solution [Carslaw and Jaeger (1959), p.378]

$$W(r,\theta,t) = \frac{1}{2\pi} \sum_{n=-\infty}^{\infty} \int_0^{\infty} \cos\{n(\theta-\theta')\}\ \bar{w}\ e^{-K\bar{w}^2(t-t')}\ \frac{\bar{U}_n(\bar{w}r')\ \bar{U}_n(\bar{w}r)\ d\bar{w}}{[J_n^2(\bar{w}r_o)+Y_n^2(\bar{w}r_o)]}$$

$$r \geqslant r_o,\ 0 \leqslant \theta \leqslant 2\pi,\ t \geqslant 0 \qquad - (22a)$$

where

$$\bar{U}(\bar{w}r) = J_n(\bar{w}r)\ Y_n(\bar{w}r_o) - J_n(\bar{w}r_o)\ Y_n(\bar{w}r)$$

This solution was also presented in a slightly different form by Carslaw (1921) p.196, namely

$$W(r,\theta,t) = \frac{1}{4\pi} \sum_{n=-\infty}^{\infty} \int_{-\infty}^{\infty} \cos\{n(\theta-\theta')\}\ \bar{w}\ e^{-K\bar{w}^2(t-t')}\ x \qquad - (22b)$$

$$x\ \frac{H_n^{(1)}(\bar{w}r')}{H_n^{(1)}(\bar{w}r_o)}\ \{J_n(\bar{w}r)\ H_n^{(1)}(\bar{w}r_o) - J_n(\bar{w}r_o)\ H_n^{(1)}(\bar{w}r)\}\ d\bar{w}$$

for $r \geqslant r_o$, $0 \leqslant \theta \leqslant 2\pi$, $t \geqslant 0$. To obtain the elliptic solution we reverse the 'degeneration' procedure of section 3.2.1 and determine what equation produces a 'circular limit' given by (22a). Making the change of variable $\bar{w} = (\rho/K)^{\frac{1}{2}}$ then $\bar{w}\ d\bar{w} = d\rho/2K$ and (22a) becomes

$$W(r,\theta,t) = \sum_{n=-\infty}^{\infty} \int_0^{\infty} \frac{\cos\{n(\theta-\theta')\}}{4\pi K}\ e^{-\rho(t-t')}\ \frac{\bar{U}_n[(\rho/K)^{\frac{1}{2}}r']\ \bar{U}_n[(\rho/K)^{\frac{1}{2}}r]\ d\rho}{J_n^2[(\rho/K)^{\frac{1}{2}}r_o] + Y_n^2[(\rho/K)^{\frac{1}{2}}r_o]}$$

(equation 23) for $r \geqslant r_o$, $0 \leqslant \theta \leqslant 2\pi$, $t \geqslant 0$. Now, let $\omega = h^2\rho/4K$ ($d\rho = 4K\omega/h^2$) and (23) is the limiting case with $h \to 0$ ($\omega \to 0$) of the equation

$$W(\xi,\eta,t) = \frac{1}{\pi h^2} \sum_{n=-\infty}^{\infty} \int_0^{\infty} \frac{e^{-\alpha\omega(t-t')}}{(p_n')^2} \frac{g_n(\eta;\omega)\, f_n(\xi';\omega)\, f_n(\xi;\omega)}{[Ce_n^2(\xi_o,\omega) + Fey_n^2(\xi_o,\omega)]}\, d\omega \quad - (24)$$

for $\xi \geqslant \xi_o$, $0 \leqslant \eta \leqslant 2\pi$, $t \geqslant 0$ and where, for convenience, $\alpha = 4K/h^2$ and

$$f_n(\xi;\omega) = Ce_n(\xi,\omega)\, Fey_n(\xi_o,\omega) - Ce_n(\xi_o,\omega)\, Fey_n(\xi,\omega)$$
$$g_n(\eta;\omega) = ce_n(\eta,\omega)\, ce_n(\eta',\omega) + se_n(\eta,\omega)\, se_n(\eta',\omega)$$

Equation (24) gives the unit point source solution in the region bounded internally by the ellipse $\xi = \xi_o$ with a zero surface temperature. We now justify this solution before continuing with the moving source solution and begin by showing that (24) is infact a solution of the conduction equation (20). Using (24) the right-hand side of (20) is

$$-\frac{2(\cosh 2\xi - \cos 2\eta)}{\pi h^2} \sum_{n=-\infty}^{\infty} \int_0^{\infty} \frac{\omega e^{-\alpha\omega(t-t')}}{(p_n')^2} \frac{g_n(\eta;\omega)\, f_n(\xi';\omega)\, f_n(\xi;\omega)}{[Ce_n^2(\xi_o,\omega) + Fey_n^2(\xi_o,\omega)]}\, d\omega$$

(equation 25). The left-hand side of (20), using (24), yields

$$\frac{1}{\pi h^2} \sum_{n=-\infty}^{\infty} \int_0^{\infty} \frac{e^{-\alpha\omega(t-t')}}{(p_n')^2} f_n(\xi';\omega) \frac{[g_n(\eta;\omega)\, f_n''(\xi;\omega) + g_n''(\eta;\omega)\, f_n(\xi;\omega)]}{[Ce_n^2(\xi_o,\omega) + Fey_n^2(\xi_o,\omega)]}\, d\omega$$

(equation (i)) where

$$f_n'' = \frac{d^2 f}{d\xi^2} = \frac{d^2 Ce_n(\xi,\omega)}{d\xi^2} Fey_n(\xi_o,\omega) - Ce_n(\xi_o,\omega) \frac{d^2 Fey_n(\xi,\omega)}{d\xi^2} \quad - (ii)$$

and

$$g_n'' = \frac{d^2 g}{d\eta^2} = \frac{d^2 ce_n(\eta,\omega)}{d\eta^2} ce_n(\eta',\omega) + \frac{d^2 se_n(\eta,\omega)}{d\eta^2} se_n(\eta',\omega) \quad - (iii)$$

Now, $Ce_n(\xi,\omega)$ and $Fey_n(\xi,\omega)$ are solutions of the equation $x'' - (a - 2\omega\cosh 2\xi)x = 0$ and $ce_n(\eta,\omega)$ and $se_n(\eta,\omega)$ are solutions of the equation $y'' + (a - 2\omega\cos 2\eta)y = 0$. By simple manipulation (ii) and (iii) become

$$\frac{d^2 f}{d\xi^2} = (a - 2\omega\cosh 2\xi) f$$

$$\frac{d^2 g}{d\eta^2} = -(a - 2\omega\cos 2\eta) g$$

and (i) may be rearranged to

$$\sum_{n=-\infty}^{\infty} \int_0^{\infty} \frac{e^{-\kappa\omega(t-t')}}{(p_n')^2} \frac{f_n(\xi';\omega)}{\pi h^2} \cdot \frac{-2\omega(\cosh 2\xi - \cos 2\eta)\, f_n(\xi;\omega)\, g_n(\eta;\omega)\, d\omega}{[Ce_n^2(\xi_o,\omega) + Fey_n^2(\xi_o,\omega)]}$$

which is identical to equation (25). Thus we have shown that (24) is a solution of the elliptic conduction equation.

We now require to check that (24) satisfies the various initial and boundary conditions (21a) - (21c). Clearly the condition of zero temperature on the bounding ellipse (equation (21a)) is satisfied since $f_n(\xi_o;\omega) \equiv 0$. Similarly, since $\lim_{\xi \to \infty} Ce_n(\xi,\omega) = 0$ and $\lim_{\xi \to \infty} Fey_n(\xi,\omega) = 0$ then $\lim_{\xi \to \infty} f_n(\xi;\omega) = 0$ and so condition (21b) is satisfied.

To satisfy (21c) we see that we require $\lim_{t \to t'} W(\xi,\eta,t) = 0$.

Assuming that the integral is a continuous function of t at $t = t'$ then we require to show that

$$\sum_{n=-\infty}^{\infty} \int_0^{\infty} \frac{g_n(\eta;\omega)\, f_n(\xi';\omega)\, f_n(\xi;\omega)\, d\omega}{(p_n')^2 [Ce_n^2(\xi_o,\omega)+Fey_n^2(\xi_o,\omega)]} = 0$$

for all ξ and η except at (ξ',η') where the temperature is infinite. At this point there should exist a source of unit strength. This may be shown by integrating over the whole solution region to find the heat liberated and dividing the answer by ρc. To obtain the above limiting requirement the Mathieu functions are first expressed in terms of Bessel function series. By considering suitable contour integrals the required result may be obtained.

3.3.2 The Differential Line Source

Having established that (24) is the Green's function for a unit point source at (ξ',η',t') we now consider the interface curve $\xi = S(\eta,t)$ to represent a line source of heat moving through the solution region $\xi > \xi_o$, $0 \leqslant \eta \leqslant 2\pi$, $t > 0$. Following Rathjen and Jiji (1971) we see that an element $d\eta'$ located at η' contains a source of length

$$ds = \{1 + [\frac{\partial S}{\partial \eta'}(\eta',t')]^2\}^{\frac{1}{2}} d\eta'$$

at $t = t'$. The normal velocity of this line element is

$$\frac{\partial S}{\partial t'}(\eta',t') \{1 + [\frac{\partial S}{\partial \eta'}(\eta',t')]^2\}^{-\frac{1}{2}}$$

and the element ds covers the area $\frac{\partial S}{\partial t'} d\eta' dt'$ in a time dt' absorbing $\rho L/(T_e - T_o) \frac{\partial S}{\partial t'} d\eta' dt'$ units of heat. Since by definition the strength of a source is equal to the number of units of heat divided by ρc then the strength of the differential source at the point $(S(\eta', t'), \eta', t')$ is $\phi \frac{\partial S}{\partial t'} (\eta', t') d\eta' dt'$. ϕ is the latent heat parameter $L/[c(T_e - T_o)]$. By summing over all the differential sources the function $V(\xi, \eta, t)$ may be determined. Re-iterating that $W(\xi, \eta, t)$, equation (24), is the solution for a unit point source at (ξ', η', t') with zero surface temperature and zero initial temperature then the temperature distribution due to the differential source at $(S(\eta', t'), \eta', t')$ is

$$\phi \frac{\partial S}{\partial t'} (\eta', t') \frac{d\eta' dt'}{\pi h^2} \sum_{n=-\infty}^{\infty} \int_0^{\infty} \frac{e^{-\alpha\omega(t-t')} g_n(\eta;\omega) f_n(S(\eta', t');\omega) f_n(\xi;\omega) d\omega}{(p_n')^2 \quad [Ce_n^2(\xi_o,\omega) + Fey_n^2(\xi_o,\omega)]}$$

where f_n and g_n have been previously defined. To obtain $V(\xi, \eta, t)$ we integrate the differential solution along the interface, that is, with respect to η' $(0 \leqslant \eta' \leqslant 2\pi)$ and t' $(0 \leqslant t' \leqslant t)$. Hence $V(\xi, \eta, t)$ is given by

$$V(\xi, \eta, t) = \frac{\phi}{\pi h^2} \int_0^t \int_0^{2\pi} \frac{\partial S}{\partial t'} (\eta', t') \, x \qquad - (26)$$

$$x \sum_{n=-\infty}^{\infty} \int_0^{\infty} \frac{e^{-\alpha\omega(t-t')}}{(p_n')^2} \frac{g_n(\eta;\omega) f_n(S(\eta', t');\omega) f_n(\xi;\omega)}{[Ce_n^2(\xi_o,\omega) + Fey_n^2(\xi_o,\omega)]} d\omega \, d\eta' dt'$$

Rathjen and Jiji (1971) managed to simplify their corresponding expresion by using the similarity in the co-ordinate system to reduce

the number of integration variables. This is not possible for the elliptic problem although the change of integration variable $\tau = t'/t$ removes the indefinite integration with respect to t' and (26) becomes

$$V(\xi,\eta,t) = \frac{\phi}{\pi h^2} \int_0^1 \int_0^{2\pi} \frac{\partial S(\eta',\tau)}{\partial \tau} \times \qquad - (27)$$

$$\times \sum_{n=-\infty}^{\infty} \int_0^{\infty} \frac{e^{-\alpha\omega t(1-\tau)} \; g_n(\eta;\omega) \; f_n(S(\eta',\tau);\omega) \; f_n(\xi;\omega)}{(p_n')^2 \quad [Ce_n^2(\xi_o,\omega) + Fey_n^2(\xi_o,\omega)]} \, d\omega \, d\eta' d\tau$$

From (19d), (9) and (27) we obtain the following integro-differential equation for the interface $S(\eta,t)$

$$\frac{T_f - T_o}{T_e - T_o} = 1 + \frac{2}{\pi} \sum_{m=0}^{\infty} \int_0^{\infty} \frac{e^{-\alpha\omega t} \; A_0^{(2m)} \; G_{2m}(S(\eta,t),\omega) \; ce_{2m}(\eta,\omega)}{[Ce_{2m}^2(\xi_o,\omega) + Fey_{2m}^2(\xi_o,\omega)]} \, d\omega$$

$$+ \frac{\phi}{\pi h^2} \int_0^1 \int_0^{2\pi} \frac{\partial S(\eta',\tau)}{\partial \tau} \times \qquad - (28)$$

$$\times \sum_{n=-\infty}^{\infty} \int_0^{\infty} \frac{e^{-\alpha\omega t(1-\tau)} \; g_n(\eta;\omega) \; f_n(S(\eta',\tau);\omega) \; f_n(S(\eta,t);\omega)}{(p_n')^2 \quad [Ce_n^2(\xi_o;\omega) + Fey_n^2(\xi_o;\omega)]} \, d\omega \, d\eta' d\tau$$

To determine the corresponding expression for a 'flat plate' heating element we consider the expression (28) as the eccentricity of the bounding ellipse tends to unity and $\xi_o \to 0$ and obtain

$$\frac{T_f - T_o}{T_e - T_o} = 1 + \frac{2}{\pi} \sum_{m=0}^{\infty} \int_0^{\infty} \frac{e^{-\alpha\omega t} A_0^{(2m)} G_{2m}(S(\eta,t),\omega)\ ce_{2m}(\eta,\omega)\ d\omega}{[Ce_{2m}^2(0,\omega) + Fey_{2m}^2(0,\omega)]}$$

$$+ \frac{\phi}{\pi h^2} \int_0^1 \int_0^{2\pi} \frac{\partial S(\eta',\tau)}{\partial \tau} \times \qquad - (29)$$

$$\times \sum_{n=-\infty}^{\infty} \int_0^{\infty} \frac{e^{-\alpha\omega t(1-\tau)}\ g_n(\eta;\omega)\ f_n(S(\eta',\tau);\omega)\ f_n(S(\eta,t);\omega)}{(p_n')^2\ [Ce_n^2(0,\omega) + Fey_n^2(0,\omega)]}\ d\omega\ d\eta' d\tau$$

3.4 The Numerical Evaluation of Mathieu Functions

The solution of equation (29) gives the interface location at any time t. However, because of the complexity of the expression an analytic solution is virtually impossible and even a numerical solution is hard to obtain. Part of the difficulty lies in the fact that the interface $S(\eta,t)$ is a parameter of the functions G_{2m} and f_n. Also the evaluation of the functions G_{2m}, g_n and f_n ordinarily would cause difficulty since each is a function of Mathieu functions which themselves are infinite series of Bessel functions. The optimum expressions for Mathieu functions (from a computational point of view) are given in chapter 13 of McLachlan's book (1947) and consist of infinite Bessel function product series. The advantage of this formulation is a more rapid convergence of the series, particularly for large arguments. Each term in the series is also multiplied by a suitable characteristic co-efficient $A_{2r}^{(2n)}$, $A_{2r+1}^{(2n+1)}$, $B_{2r+2}^{(2n+2)}$ or

$B_{2r+1}^{(2n+1)}$ where n is the order of the Mathieu function and r signifies the r^{th} term in the infinite product series. These co-efficients are in turn dependent on the characteristic numbers a_{2n}, a_{2n+1}, b_{2n+2} and b_{2n+1}. As noted in section 3.2 these represent the legitimate values of the separation constant 'a' in the cannonical forms of Mathieu's equation (2d) and modified equation (2c). Their evaluation is non-trivial and is described in chapter 3 of McLachlan's book (1947).

In concluding this chapter we have successfully utilized Lightfoot's (1930) method of 'moving sources' to provide an analytic representation of the thermal phase-change solution in the region surrounding a heated element. This solution is a particular case of the more general problem of melting/freezing outside an infinite elliptic cylinder. However, because of the complexity of the solution, the numerical evaluation of the expression would appear to be as involved as obtaining the solution its self and as yet no suitable codes have been designed to compute the Mathieu functions concerned. It is hoped to remedy this situation in the future.

CHAPTER IV

AN INDUSTRIAL FREEZING PROBLEM

PHASE CHANGE IN A CONICAL CYLINDER WITH A FLUID EXHIBITING A PERIODIC VELOCITY PERTURBATION

4.1 Introduction

Up to this point we have been concerned with phase change problems in which it has been assumed that there is no fluid motion in the liquid phase. The introduction of a non-zero velocity distribution considerably complicates the problem in several ways. In the first instance we require an extra equation to describe the velocity profile at any point in the solution region. A non-zero velocity also produces an extra heat transfer term in the heat balance equation (at the freezing front) due to convection. Convection also permeates the whole liquid solution region. This is reflected by the energy equation (as opposed to the conduction equation) which now governs the temperature distribution in the liquid and expresses the dependence of the temperature on the fluid velocity.

The solidification of a liquid flowing through a pipe is a situation common to a variety of industrial processes [for example, Saito (1921), Szekely and Lee (1968), Massey and Sunderland (1972), Abuaf and Gutfinger (1973), Moalem et al (1973), Ehrich et al (1978)] and in some cases may be a critical consideration. In this chapter we pay particular attention to the solidification of a liquid flowing through a pipe and describe a numerical investigation into one such problem, that of slag solidification in a "fixed bed" gasification

process. The groundwork for this problem was completed at the British Gas Corporation's Midlands Research Station in the summer of 1981 and appeared in an internal report.

The class of Stefan problems concerned with flow through pipes has been studied for some time. The seminal publication of Zerkle and Sunderland (1968) examines the pressure drop along a parallel cylinder (of circular cross-section) necessary to maintain a constant inlet velocity as a solid crust forms on the inside of the tube wall. In common with much of the work in this area they assume laminar fluid flow, neglect axial heat conduction and solve a pseudo-steady-state problem (in their case an extended Graetz (1885) problem). The explicit time dependence of the liquid temperature is also neglected and a steady-state solution is presented for the solid phase. The model was extended to include transient problems by Ozisik and Mulligan (1969). They proposed slug-flow velocity profile and used Hankel and Laplace transforms to reduce the problem to a set of ordinary differential equations. The situation in which blockage might occur was considered by Des Ruisseaux and Zerkle (1969) and more recently by Sampson and Gibson (1981). They use the Graetz problem to derive an expression for the radial temperature gradient at the solid/liquid interface which is required to calculate the interface velocity. In a subsequent paper Sampson and Gibson (1982) extend their model to include turbulent flow.

In this investigation we use a laminar approximation for the velocity profile since the fluid velocities considered are fairly small, $0 < V_{bulk} < 0.1\ ms^{-1}$ (=> low Reynolds numbers and hence ~laminar flow). The model also incorporates two additional features which are both considered to give a more realistic approximation to

the following freezing problem. We first propose to extend the model from that of a circular cylinder to the more general cone (of circular cross-section) as this provides a reasonable geometric approximation. Secondly we consider a pulsating fluid motion as opposed to the usual uni-directional flow. This models observed effects in the freezing problem. Having formulated the problem in terms of the governing partial differential equations a co-ordinate transformation [for example Duda et al (1975)] is used to immobilise the freezing front (for computational convenience).

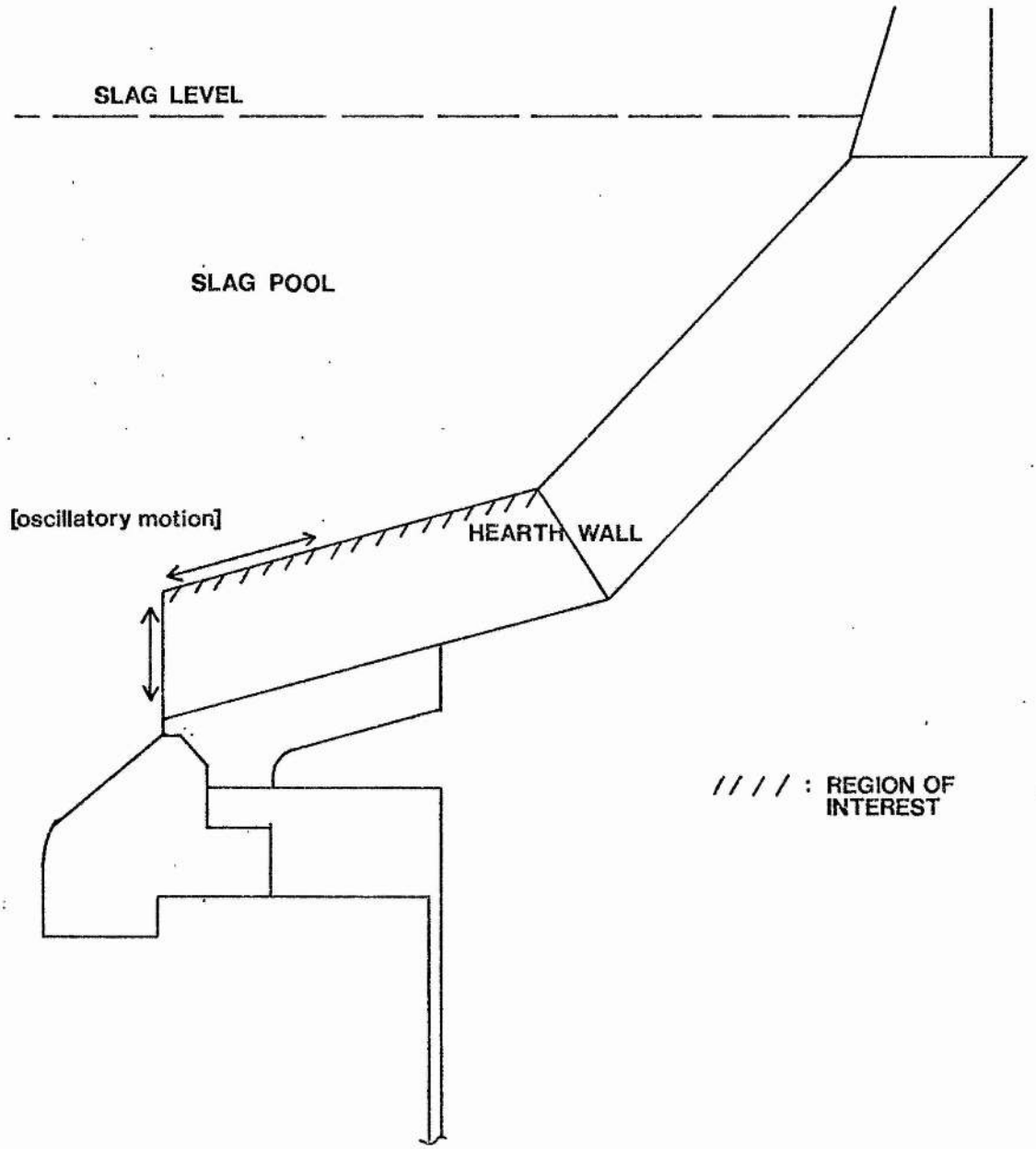
4.2 The Physical Problem

The slagging gasifier (see figure 4.1) forms part of the process known as "fixed bed" gasification [see, for example, Tart and Rampling (1980)]. Near the bottom of the hearth a mixture of steam and oxygen (air) enters the gasifier and individual lumps of fuel (for example coal) are progressively gasified, coal ash being formed as one product of the process. If the steam to oxygen ratio is sufficiently low (1/2 vol./vol.) the combustion temperature in the fuel bed exceeds the fusion temperature of coal ash and the ash becomes liquid slag which accumulates in the slag pool. During this process a jet of hot gases is passed up through the slag tap and into the slag pool. This jet helps to keep the slag molten and causes circulation in the liquid slag. In addition the jet aids the prevention of 'dribbling'. At intervals the jet is turned down and liquid slag is tapped into the quench chamber. We are concerned with the period between tapping.

During this time the slag pool wall is maintained at a temperature below the fusion temperature of slag and a crust of solid

Figure 4.1

Gasifier Slag Pool



slag is present on the wall. This crust serves a purpose in so far as it helps protect the hearth wall from erosion due to the circulating slag. The erosion effects are similar to those occurring in a blast furnace used in the production of iron, a problem that has been tackled, using finite differences, by Yoshikawa and Szekely (1981). However, the thickness of the crust must remain below a critical value to avoid blockage, particularly in the region of the slag tap.

The jet of hot gases does not enter the slag pool as a continuous stream but as a series of bubbles. Near the bottom of the slag pool these bubbles cause a localised oscillatory motion to be super-imposed on the overall circulation pattern which displays a net 'downward' movement. This effect has been observed in experiments conducted on a model gasifier at the British Gas Corporations' Midlands Research Station. One aim of this investigation is to model the pulsating behaviour and to determine what effect, if any, the rate of slag pulsation has on the growth of the solid crust. It will also be possible to examine the effect that varying 'shapes' of the cylinder have on the crust formation.

4.3 A Sensitivity Analysis

As a first step towards understanding the hearth slag pool problem a semi-infinite one-dimensional freezing problem is considered. It is the aim of this analysis to obtain an appreciation of the properties of slag and to perform a very simple sensitivity analysis on various variables associated with this problem. We also consider the effect of neglecting the change in volume due to solidification. This problem is in no way designed to model the slag

pool situation.

Since the hearth slag pool problem considers a constant maintained temperature on the slag pool wall we shall use the classical Stefan problem as the basis for this analysis. That is, we consider a semi-infinite region $x \in [0,\infty)$ which contains liquid slag initially at a temperature T_o ($>T_f$, the fusion temperature). At time $t = 0.0$ the plane $x = 0.0$ is subject to a temperature T_w ($<T_f$) which is subsequently maintained. Consequently a solid/liquid interface propagates along the positive x-axis. This problem is well documented and, for the case of no change of volume on solidification, has the analytic solution [see Carslaw and Jaeger (1959), pp.283-286]

$$T_S(x,t) = T_w + (T_f - T_w)\,\frac{\mathrm{erf}[x/2(\kappa_S t)^{\frac{1}{2}}]}{\mathrm{erf}(\lambda)} \qquad - (1a)$$

$$0 \leqslant x \leqslant x_o(t)\ ,\ t \geqslant 0.0$$

for the solid phase temperature distribution and

$$T_L(x,t) = T_o - (T_o - T_f)\,\frac{\mathrm{erfc}[x/2(\kappa_L t)^{\frac{1}{2}}]}{\mathrm{erfc}[\lambda(\kappa_S/\kappa_L)^{\frac{1}{2}}]} \qquad - (2a)$$

$$x \geqslant x_o(t)\ ,\quad t \geqslant 0.0$$

for the liquid phase temperature. κ_S and κ_L are the solid and liquid thermal diffusivities respectively. The solid/liquid interface location $x_o(t)$ is given by

$$x_o(t) = 2\lambda(\kappa_S t)^{\frac{1}{2}}\ ,\quad t \geqslant 0.0 \qquad - (3a)$$

The freezing parameter λ is the root of the transcendental equation

$$\frac{e^{-\lambda^2}}{\operatorname{erf}(\lambda)} - \frac{k_L \kappa_S^{\frac{1}{2}}}{k_S \kappa_L^{\frac{1}{2}}} \frac{(T_o - T_f)}{(T_f - T_w)} \frac{e^{-\lambda^2(\kappa_S/\kappa_L)}}{\operatorname{erfc}[\lambda(\kappa_S/\kappa_L)^{\frac{1}{2}}]} = \frac{L \lambda \pi^{\frac{1}{2}}}{c(T_f - T_w)} \qquad - (4a)$$

where L is the latent heat of fusion and k_S and k_L are the solid and liquid thermal conductivities respectively.

The solution given by equations (1a) to (4a) is based on the assumption that there is no change of volume on solidification (despite the possibility that $\rho_s \neq \rho_L$). If we incorporate a volume change there will be motion of the liquid given by $u_x = -(\rho_S/\rho_L - 1)\frac{dx_o}{dt}$. The corresponding solution is [Carslaw and Jaeger (1959), pp. 290-291]

$$T_S(x,t) = T_w + (T_f - T_w) \frac{\operatorname{erf}[x/2(\kappa_S t)^{\frac{1}{2}}]}{\operatorname{erf}(\lambda)} \qquad - (1b)$$

$$0 \leqslant x \leqslant x_o(t) \ , \ t \geqslant 0.0$$

$$T_L(x,t) = T_o - (T_o - T_f) \frac{\operatorname{erfc}[x/2(\kappa_L t)^{\frac{1}{2}} + (\rho_S/\rho_L - 1)(\kappa_S/\kappa_L)^{\frac{1}{2}}]}{\operatorname{erfc}[\lambda(\kappa_S/\kappa_L)^{\frac{1}{2}}(\rho_S/\rho_L)]} \qquad - (2b)$$

$$x \geqslant x_o(t) \ , \ t \geqslant 0.0$$

$$x_o(t) = 2\lambda(\kappa_S t)^{\frac{1}{2}} \ , \quad t \geqslant 0.0 \qquad - (3b)$$

where the freezing parameter λ is now the root of the equation

$$\frac{e^{-\lambda^2}}{\operatorname{erf}(\lambda)} - \frac{k_L \kappa_S^{\frac{1}{2}}}{K_S \kappa_L^{\frac{1}{2}}} \frac{(T_o - T_f)}{(T_f - T_w)} \frac{e^{-\lambda^2(\kappa_S/\kappa_L)(\rho_S/\rho_L)^2}}{\operatorname{erfc}[\lambda(\kappa_S/\kappa_L)^{\frac{1}{2}}(\rho_S/\rho_L)]} = \frac{L \lambda \pi^{\frac{1}{2}}}{c(T_f - T_w)} \qquad - (4b)$$

The following analysis is conducted with respect to the solutions (1a) - (4a) and (1b) - (4b) which allows us to observe the effect of neglecting the volume change.

The following input variables are considered: T_o (initial liquid slag temperature), T_f (fusion temperature), T_w (wall temperature), ρ_S (solid density), ρ_L (liquid density), L (latent heat of fusion) and c (specific heat capacity). The effect of changing these values is to alter the value of λ calculated from equations (4a) and (4b) and hence alter $x_o(t)$ as given by equations (3a) and (3b).

The values T_o, T_f and T_w appear in (4a) and (4b) explicitly and implicitly since they may be used to estimate the bulk thermal conductivities and ultimately the diffusivities of the liquid and solid phases. From Hoy, Roberts and Wilkins (1965) the thermal conductivity is calculated according to the formulae

$$\begin{aligned} k &= 0.65\ (1 + 0.7\times10^{-3}T) && \text{Btu/ft h}^{\circ}\text{C} && T < 1000\ ^{\circ}\text{C} \\ k &= 1.12\ (1 + 2\times10^{-3}(T-1000)) && \text{Btu/ft h}^{\circ}\text{C} && T > 1000\ ^{\circ}\text{C} \end{aligned} \qquad - (4c)$$

where the conversion factor to $J/ms^{\circ}C$ is approximately 0.96089239.

The solid and liquid density ranges, as given by Hoy et al (1965), appear explicitly in equations (4a) and (4b) and also implicitly from the relation $\kappa = k/(\rho c)$. The range of latent heat considered (2×10^5 J/kg to 4×10^5 J/kg) includes most of the available estimates.

Also from Hoy et al (1965) we have a list (table 4.1) of the heat content H (in units of BTU/lb) for the temperature range $T = 1000^{\circ}C(100^{\circ}C)1700^{\circ}C$.

Table 4.1

Heat content of slag as a function of temperature.

Temperature (^{o}C)	Heat Content (BTU/lb)
1100.0	475.0
1200.0	550.0
1300.0	630.0
1400.0	710.0
1500.0	790.0
1600.0	865.0
1700.0	935.0

Using the conversion factor 1 BTU/lb $\simeq$ 2324.4839 J/kg a linear least squares approximation to the data for H(T) yields

$$H(T) \simeq 1801.48T - 876662.41 \quad J/kg$$

From, for example, Birtwhistle (1927) pp.12-13 the specific heat c is given by $c = dH/dT \simeq 1801.48$ J/kg. That is, we are assuming that c is constant for all temperatures and that $c_S = c_L \simeq 1801.48$ J/kg. Table 4.2 gives a list of the fixed data values used in the various sensitivity calculations.

Table 4.2

Standard Data

Initial Liquid Temperature	T_o	: 1400^{o}C
Fusion Temperature	T_f	: 1300^{o}C
Wall Temperature	T_w	: 100^{o}C

Solid Density	ρ_S	:	2562 kg/m^3
Liquid Density	ρ_L	:	2242 kg/m^3
Latent Heat of Fusion	L	:	2×10^5 J/kg
Specific Heat Capacity	c	:	1801.48 $J/kg^{\circ}C$

The results are presented in figures 4.2 to 4.8 for a time of 15 minutes (a listing of the computer code is given in appendix 4.1). The solid and broken lines represent the solutions neglecting and including the volume change respectively. From figure 4.7 it may be seen that the interface location is approximately linearly dependent on the latent heat, the gradient of the line varying from -1.97×10^{-5} mm/(J/kg) to -1.3×10^{-5} mm/(J/kg) [-1.88×10^{-5} mm/(J/kg) to -1.26×10^{-5} mm/(J/kg) for change of volume], and that this dependence is weak. The change in interface location over the range of latent heat considered is of the order of 10%. Thus, while it is important to get the correct 'order' of latent heat small errors are going to have little effect on the interface.

Similarly, the dependence on the solid and liquid densities (figures 4.5 and 4.6) is roughly linear, the gradients varying from -4.83×10^{-3} mm/(kg/m^3) to -4.0×10^{-3} mm/(kg/m^3) [-4.76×10^{-3} mm/(kg/m^3) to -4.34×10^{-3} mm/(kg/m^3)] and -1.54×10^{-3} mm/(kg/m^3) to -1.47×10^{-3} mm/(kg/m^3) [-1.03×10^{-3} mm/(kg/m^3) to -1.02×10^{-3} mm/(kg/m^3)] respectively. The overall change in interface location is of the order of 1% - 3%. Thus errors in the density estimates will have little effect on the interface.

A similar conclusion may be reached for the dependence on the wall temperature (figure 4.4). The overall change in interface location is ~1% with the gradient varying from -7.62×10^{-3} mm/$^{\circ}$C to

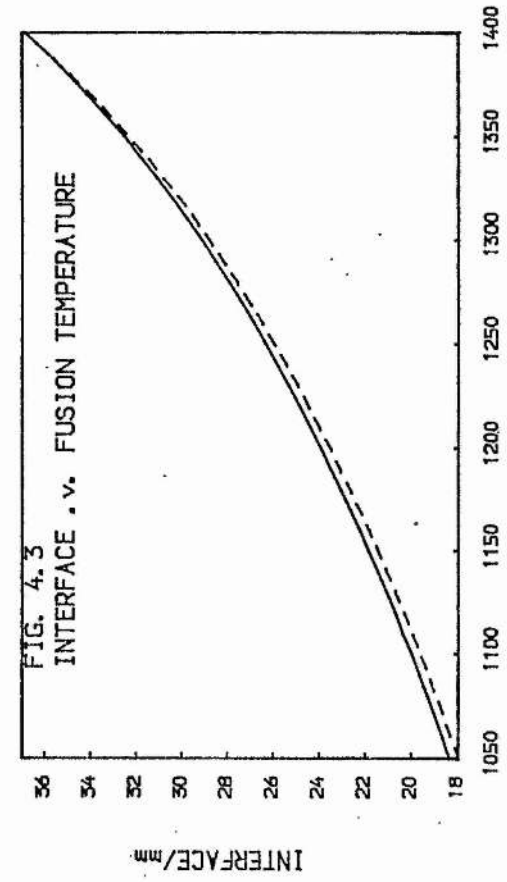


FIG. 4.3
INTERFACE . v. FUSION TEMPERATURE
INTERFACE/mm
TEMPERATURE/°C

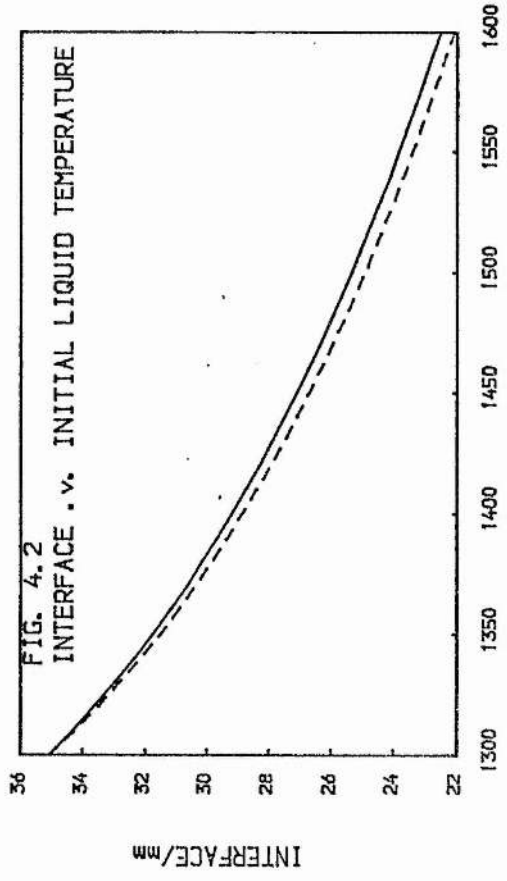


FIG. 4.2
INTERFACE . v. INITIAL LIQUID TEMPERATURE
INTERFACE/mm
TEMPERATURE/°C

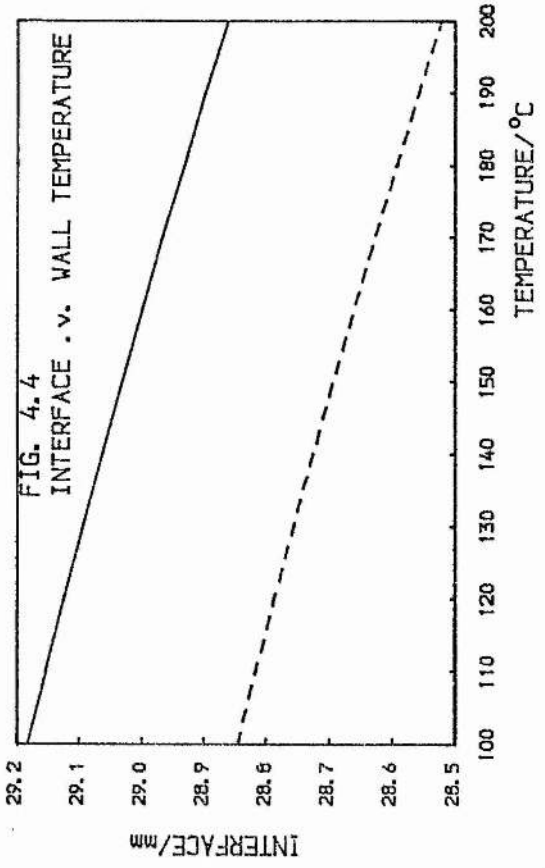


FIG. 4.4
INTERFACE . v. WALL TEMPERATURE
INTERFACE/mm
TEMPERATURE/°C

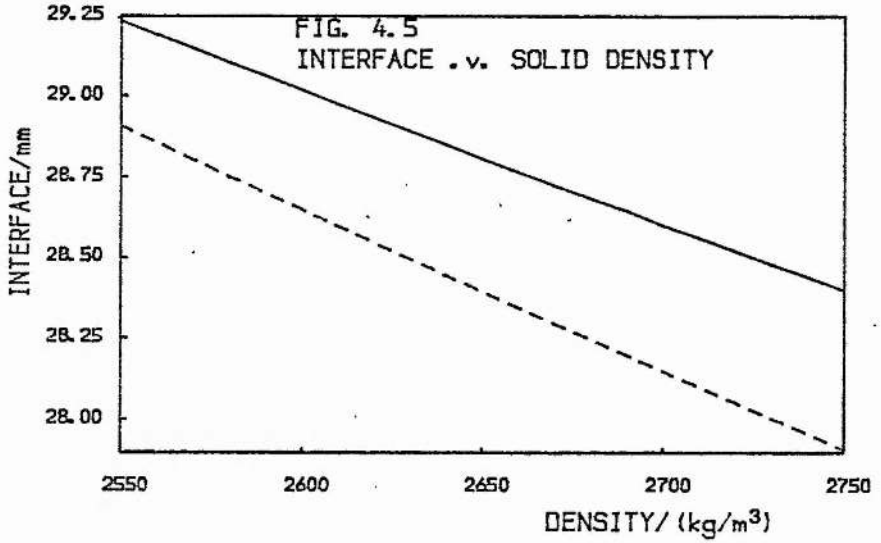


FIG. 4.5
INTERFACE . v. SOLID DENSITY
INTERFACE/mm
29.25
29.00
28.75
28.50
28.25
28.00
2550
2600
2650
2700
2750
DENSITY/ (kg/m3)

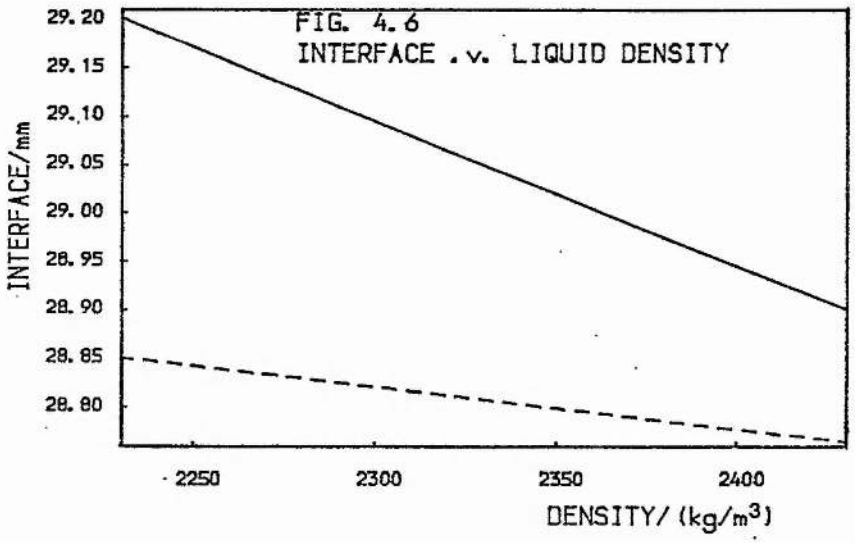


FIG. 4.6
INTERFACE . v. LIQUID DENSITY
INTERFACE/mm
29.20
29.15
29.10
29.05
29.00
28.95
28.90
28.85
28.80
2250
2300
2350
2400
DENSITY/ (kg/m3)

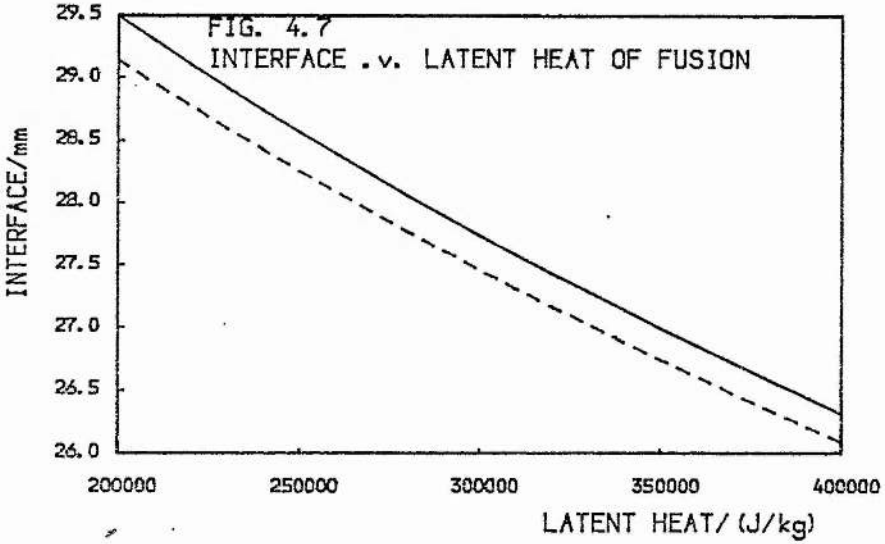


FIG. 4.7
INTERFACE . v. LATENT HEAT OF FUSION
INTERFACE/mm
29.5
29.0
28.5
28.0
27.5
27.0
26.5
26.0
200000
250000
300000
350000
400000
LATENT HEAT/ (J/kg)

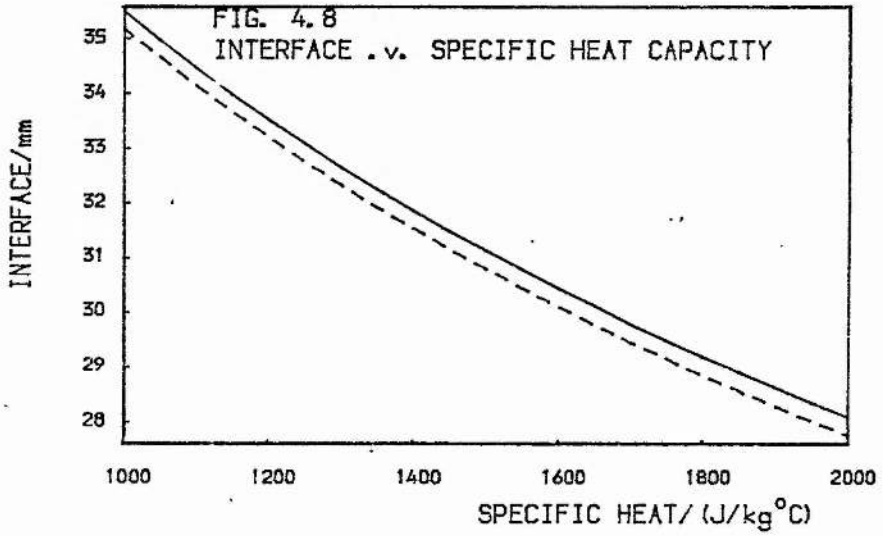


FIG. 4.8
INTERFACE . v. SPECIFIC HEAT CAPACITY
INTERFACE/mm
35
34
33
32
31
30
29
28
1000
1200
1400
1600
1800
2000
SPECIFIC HEAT/ (J/kg°C)

-6.86×10^{-3} mm/°C [-7.61×10^{-3} mm/°C to -6.85×10^{-3} mm/°C].

However, the interface would appear consistently, and appreciably, sensitive to changes in the fusion temperature (figure 4.3) and the initial liquid temperature (figure 4.2). The gradient varies from 8.73×10^{-2} mm/°C to 2.77×10^{-2} mm/°C [9.44×10^{-2} mm/°C to 2.75×10^{-2} mm/°C] for T_f and from -7.87×10^{-2} mm/°C to -2.51×10^{-2} mm/°C [-8.55×10^{-2} mm/°C to -2.51×10^{-2} mm/°C] for T_o. The overall change in the interface location is ~100% and ~-37% as one moves upwards through the respective temperature ranges. The dependence (sensitivity) is greatest for values of T_f and T_o that are close.

To a lesser extent the same may be said of the dependence on the specific heat (figure 4.8). Moving up through the range of values, the interface changes by approximately 20% with the gradient varying from -1.09×10^{-2} mm/(J/kg°C) to -5.18×10^{-2} mm/(J/kg°C) for both solutions.

With regard to the effect of neglecting the change in volume due to solidification we refer to table 4.3.

Table 4.3

Percentage change in the interface location due to the neglect of the volume change on freezing

Variable		Interface Location /mm 'Include'	'Neglect'	% Change
T_o	↑	22.09	22.52	1.96
T_f	↓	18.00	18.39	2.17
T_w	↑	28.52	28.86	1.20
S	↑	27.91	28.40	1.76
L	↓	28.85	29.20	1.21
L	↓	29.14	29.49	1.20

c	↑	27.75	28.09	1.22

(where maximum variation occurs at the upper (↑)/lower (↓) range of independent variable). It is clear that little effect is produced by this neglect, the maximum change in the interface being ~2.17%. Thus in the model for the slag pool freezing problem no account will be made for volume change on freezing.

4.4 A Mathematical Model

To model the situation at the bottom of the hearth slag pool we consider a conical cylinder in which a fluid is exhibiting a periodic type of motion. Due to the symmetry about the axis (z co-ordinate) of the cylinder (no 'angular' dependence) we may consider the section presented in figure 4.9. The full mathematical description of the problem leads to a complex system of four coupled partial differential equations which relate the solid and liquid phase temperature distributions with the liquid velocity profile and and solid/liquid interface movement. To be more realistic the velocity would be modelled by a turbulent approach although the low velocities ($<0.1\ ms^{-1}$) considered here partially justify a laminar approach since the Reynolds number will be small. An analytic solution to these equations would appear totally intractable and even an iterative numerical scheme would be cumbersome. To overcome this difficulty several simplifying approximations will be presented in the following model formulation which have the effect of reducing the problem to that of solving a pair of coupled partial differential equations. Further analysis then allows us to solve for the interface location

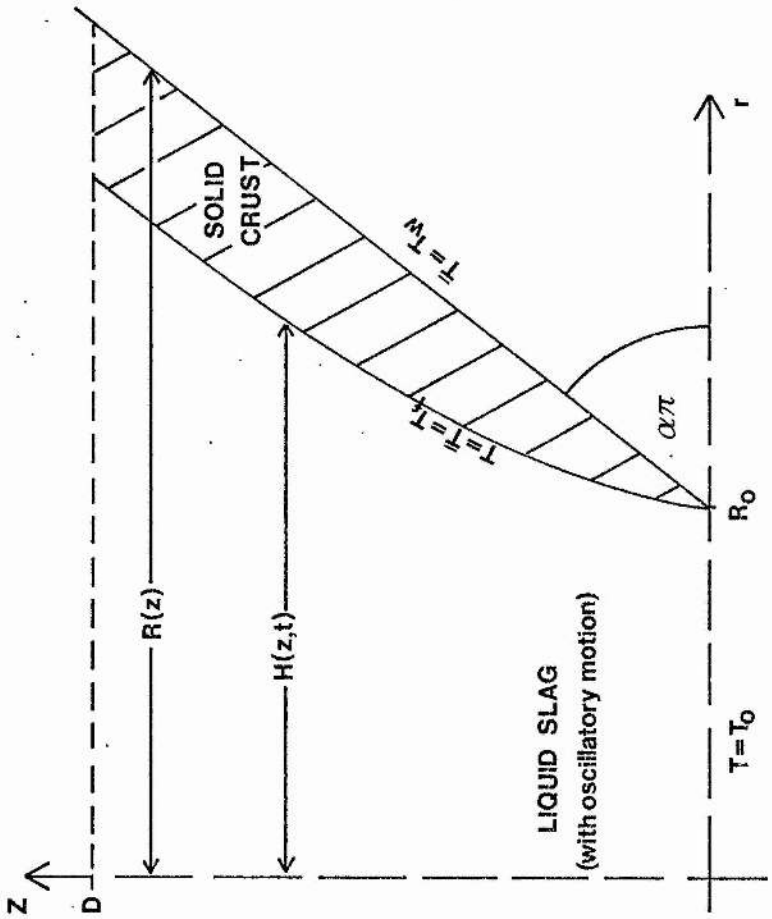


Figure 4.9

Slag Pool Model

independently and introduce a 'weak' coupling between the temperature (energy) equation and the interface.

We suppose that the material has a unique fusion temperature T_f with an associated latent heat of fusion L, density ρ_L and specific heat capacity c. At time t = 0.0 the initial temperature of the liquid is T_o ($>T_f$). At this time there is no solid phase present. For $t > 0.0$ the walls of the cylinder are held at a temperature T_w ($<T_f$) and the inlet/outlet (tap end) is held at T_o. By symmetry there is no heat flow across the z-axis. At the 'upper' end of the cylinder we assume a constant temperature gradient with respect to z. The solid/liquid interface location H(z,t) is given as the radial distance (from the z-axis) for an axial location z and time t. For $t > 0$ this interface will propagate towards the axis of the cylinder. Depending on the prevailing conditions this crust may attain a steady-state condition or blockage may occur [Sampson and Gibson (1981), (1982)].

Assuming an incompressible fluid the velocity distribution is governed by the continuity equation

$$\frac{v_r}{r} + \frac{\partial v_r}{\partial r} + \frac{\partial v_z}{\partial z} = 0 , \quad 0 \leqslant r < H(z,t), \; 0 \leqslant z \leqslant D, \; t > 0 \qquad - (5)$$

where $v_r(r,z,t)$ and $v_z(r,z,t)$ are the radial and axial velocity components respectively. D is the length (or depth) of the cylinder. For most physical problems the time scale associated with the interface movement is much longer than that associated with the velocity profile and the velocity distribution will adapt as quickly as the interface moves. With this in mind we may follow Zerkle and Sunderland (1968) and, admitting a (laminar) Poiseuille type flow, propose the pseudo-steady-state velocity profiles

$$v_r(r,z,t) = \frac{2Qr}{H^3}\frac{\partial H}{\partial z}\left[1 - (r/H)^2\right]\left[1 - \frac{1}{2}\sin(\pi\omega t)\right] \qquad - (6)$$

$$v_z(r,z,t) = \frac{2Q}{H^2}\left[1 - (r/H)^2\right]\left[1 - \frac{1}{2}\sin(\pi\omega t)\right] \qquad - (7)$$

for $0 \leqslant r \leqslant H(z,t)$, $0 \leqslant z \leqslant D$, $t \geqslant 0.0$. R_o is the inlet/outlet radius ($z = 0.0$), $Q(t)$ is the volume flux and the multiplier $1 - \frac{1}{2}\sin(\pi\omega t)$ is used to simulate the pulsating fluid motion. ω is the period of oscillation and is equal to the rate at which gas bubbles are formed and enter the slag pool. It is easily shown that equations (6) and (7) satisfy the continuity equation (5) along with a no-slip condition at the solid/liquid interface. Their use removes the necessity of computing iterated solutions for the velocity distribution at each time step required for solution. The factor of $\frac{1}{2}$ multiplying the sin term is arbitrary although for simplicity this factor should be less than unity to ensure that the 'periodic multiplier' in equations (6) and (7) is always positive. A term that was allowed to become zero or negative would change the form of the liquid energy equation, equation (13), and so lead to problems for a numerical solution. This observation was experienced in some initial numerical solution schemes tested.

In the solid phase we assume that the temperature distribution $T_S(r,z,t)$ adapts as fast as the interface moves and that radial heat conduction dominates axial heat conduction (for a thin crust this is a reasonable assumption since the radial temperature gradient is likely to dominate the axial temperature gradient). The usual heat conduction equation then reduces to the steady-state form

$$\frac{\partial^2 T_S}{\partial r^2} + \frac{1}{r}\frac{\partial T_S}{\partial r} = 0 \ , \quad H(z,t) < r < R(z), \ 0 \leqslant z \leqslant D, \ t > 0 \qquad - (8)$$

where R(z) is the equation of the cylinder wall and is given by

$$R(z) = R_o + z \cot(\alpha\pi) \ , \quad 0 \leqslant z \leqslant D \qquad - (9)$$

$\alpha\pi$ is the wall elevation in radians. Equation (8) may be solved subject to the boundary conditions

$$T_S(H(z,t),z,t) = T_f \ , \quad r = H(z,t), \ 0 \leqslant z \leqslant D, \ t > 0 \qquad - (10)$$

$$T_S(R(z) \ ,z,t) = T_w \ , \quad r = R(z) \ , \ 0 \leqslant z \leqslant D, \ t > 0 \qquad - (11)$$

Solving equation (8) yields the pseudo-steady-state solution

$$T_S(r,z,t) = \frac{T_f(\ln R - \ln r) - T_w(\ln H - \ln r)}{(\ln R - \ln H)} \ , \quad \begin{matrix} H(z,t) \leqslant r \leqslant R(z) \\ 0 \leqslant z \leqslant D \ , \ t \geqslant 0 \end{matrix} \qquad - (12)$$

The remaining pair of equations to be solved are the <u>energy equation</u> describing the liquid temperature distribution $T_L(r,z,t)$ and a <u>heat balance equation</u> governing the interface movement. From, for example, Howarth (1953) the liquid energy equation is

$$v_r \frac{\partial T_L}{\partial r} + v_z \frac{\partial T_L}{\partial z} + \frac{\partial T_L}{\partial t} = \kappa_L\left(\frac{\partial^2 T_L}{\partial r^2} + \frac{1}{r}\frac{\partial T_L}{\partial r} + \frac{\partial^2 T_L}{\partial z^2}\right) \qquad - (13)$$

$$0 \leqslant r < H(z,t) \ , \ 0 < z \leqslant D \ , \ t > 0.0$$

Using equations (6) and (7) for the velocity components $v_r(r,z,t)$ and $v_z(r,z,t)$ we obtain

$$\frac{2Q[1-(r/H)^2]}{H^2}\left[1 - \frac{1}{2}\sin(\pi\omega t)\right]\left[\frac{r}{H}\frac{\partial H}{\partial z}\frac{\partial T_L}{\partial r} + \frac{\partial T_L}{\partial z}\right] + \frac{\partial T_L}{\partial t} = \kappa_L\left(\frac{\partial^2 T_L}{\partial r^2} + \frac{1}{r}\frac{\partial T_L}{\partial r} + \frac{\partial^2 T_L}{\partial z^2}\right) \qquad - (14)$$

$$0 \leqslant r < H(z,t)\ ,\ 0 < z \leqslant D\ ,\ t > 0.0$$

The relevant initial and boundary conditions are

$$T_L(r,z,0) = T_o\ ,\quad 0 \leqslant r < R(z),\ 0 \leqslant z \leqslant D,\ t = 0.0 \qquad - (15)$$

$$T_L(r,0,t) = T_o\ ,\quad 0 \leqslant r < R_o\ ,\ z = 0.0\ ,\ t > 0.0 \qquad - (16)$$

$$T_L(H,z,t) = T_f\ ,\quad r = H(z,t)\ ,\ 0 \leqslant z \leqslant D,\ t > 0.0 \qquad - (17)$$

$$\frac{\partial T_L}{\partial r}(0,z,t) = 0\ ,\quad r = 0.0\ ,\ 0 \leqslant z \leqslant D,\ t > 0.0 \qquad - (18)$$

The usual heat balance equation at the interface expresses the amount of heat liberated (or absorbed) when the interface moves a small distance in terms of the heat flux normal to the interface in the solid and liquid phases. Due to the presence of a flowing liquid we also consider heat dissipation due to (forced) convection. If h is the convective heat transfer co-efficient (in units of $J/m^2 s^\circ C$) then the effective convective heat flux at the interface may be approximated by

$$q_c = h(T_f - T_w) \qquad - (19)$$

The interface condition then becomes [see Patel (1968)]

$$L\rho_S \frac{\partial H}{\partial t} = k_S \frac{\partial T_S}{\partial r} - k_L \frac{\partial T_L}{\partial r} + h(T_f - T_w) \qquad - (20)$$

$$r = H(z,t) \ , \ 0 \leqslant z \leqslant D \ , \ t > 0.0$$

where ρ_S is the solid density and k_S and k_L represent the solid and liquid thermal conductivities respectively. Again, axial conduction has been neglected in comparison to radial conduction. If we fix the inlet/outlet interface location at R_o then the initial and boundary conditions are

$$H(z,0) = R(z) \ , \quad 0 \leqslant z \leqslant D \ , \ t = 0.0 \qquad - (21)$$

$$H(0,t) = R_o \quad , \quad z = 0.0 \quad , \ t \geqslant 0.0 \qquad - (22)$$

We may utilize equation (12) by differentiating $T_S(r,z,t)$ with respect to r and evaluating the resulting expression at the interface $r = H(z,t)$. Substituting into (20) yields the interface condition

$$L\rho_S \frac{\partial H}{\partial t} = \frac{k_S \ (T_w - T_f)}{H(\ln R - \ln H)} - k_L \frac{\partial T_L}{\partial r} + h(T_f - T_w) \qquad - (23)$$

for $r = H(z,t)$, $0 \leqslant z \leqslant D$ and $t > 0.0$. The original problem has now been reduced to the solution of the pair of coupled equations, (14) and (23). While an analytic solution is still intractable the use of a numerical approach is now more feasible. With a view to this it is now convenient to normalise the liquid temperature distribution $T_L(r,z,t)$. Introducing the variable $U_L(r,z,t)$ by

$$U_L = \frac{T_L - T_f}{T_o - T_f}$$

yields the normalised freezing problem

$$\frac{2Q}{H^2}[1-(r/H)^2]\ [1 - \frac{1}{2}\sin(\pi\omega t)]\ [\frac{r}{H}\frac{\partial H}{\partial z}\frac{\partial U_L}{\partial r} + \frac{\partial U_L}{\partial z}] + \frac{\partial U_L}{\partial t} = \chi \qquad - (24)$$

$$\text{where } \chi = \kappa_L \left(\frac{\partial^2 U_L}{\partial r^2} + \frac{1}{r} \frac{\partial U_L}{\partial r} + \frac{\partial^2 U_L}{\partial z^2} \right)$$

for $0 \leqslant r < H(z,t)$, $0 < z \leqslant D$ and $t > 0.0$. The initial and boundary conditions become

$$U_L(r,z,0) = 1.0 , \quad 0 \leqslant r < R(z), \; 0 \leqslant z \leqslant D, \; t = 0.0 \qquad - (25)$$

$$U_L(r,0,t) = 1.0 , \quad 0 \leqslant r < R_o , \; z = 0.0 , \; t > 0.0 \qquad - (26)$$

$$U_L(H,z,t) = 0.0 , \quad r = H(z,t) , \; 0 \leqslant z \leqslant D, \; t > 0.0 \qquad - (27)$$

$$\frac{\partial U_L}{\partial r}(0,z,t) = 0.0 , \quad r = 0.0 , \; 0 \leqslant z \leqslant D, \; t > 0.0 \qquad - (28)$$

For the interface we obtain

$$L \rho_S \frac{\partial H}{\partial t} = \frac{k_S (T_w - T_f)}{H(\ln R - \ln H)} - k_L (T_o - T_f) \frac{\partial U_L}{\partial r} + h(T_f - T_w) \qquad - (29)$$

for $r = H(z,t)$, $0 \leqslant z \leqslant D$ and $t > 0.0$. The conditions become

$$H(z,0) = R(z) , \quad 0 \leqslant z \leqslant D, \; t = 0.0 \qquad - (30)$$

$$H(0,t) = R_o , \quad z = 0.0 , \; t > 0.0 \qquad - (31)$$

It is proposed to solve the pair of equations using a finite difference scheme. These schemes are best suited to regular grids constructed over the solution region. If a rectangular (or cylindrical) grid is constructed for this problem it is unlikely that at any given time the interface would lie along a series of nodes. Thus we would require to develop special difference formulae to approximate the energy equation (24) at points adjacent to the interface. A means to overcome this problem is to utilize a

co-ordinate transformation which immobilises the moving interface. The transformation can be so designed as to fix the interface to lie along a set of nodes in the new co-ordinate system. With this in mind we consider the variable

$$\lambda(r,z,t) = r/H(z,t)$$

It is easily seen that $0 \leqslant \lambda \leqslant 1.0$ for $0 \leqslant r \leqslant H(z,t)$. That is, the interface is fixed at $\lambda = 1.0$ and the new solution region is the rectangle $0 \leqslant \lambda \leqslant 1.0$, $0 \leqslant z \leqslant D$. Using elementary differential calculus and noting the identities

$$\frac{\partial\lambda}{\partial r} = \frac{1}{H}, \quad \frac{\partial^2\lambda}{\partial r^2} \quad 0.0, \quad \frac{\partial\lambda}{\partial z} = -\frac{\lambda}{H}\frac{\partial H}{\partial z}$$

$$\frac{\partial^2\lambda}{\partial z^2} = -\frac{\lambda}{H}\frac{\partial^2 H}{\partial z^2} + \frac{2\lambda}{H^2}(\partial H/\partial z)^2$$

$$\frac{\partial\lambda}{\partial t} = -\frac{\lambda}{H}\frac{\partial H}{\partial t}, \quad \frac{\partial H}{\partial\lambda} \equiv 0.0$$

it is easily shown that the problem described by (24) - (31) is mapped to the following immobilised freezing problem. For $U_L(\lambda,z,t)$ we have

$$\frac{\partial U_L}{\partial t} = -\frac{2Q(1-\lambda^2)(1-\frac{1}{2}\sin(\pi\omega t))}{H^2}\frac{\partial U_L}{\partial z} + \frac{\lambda}{H}\frac{\partial H}{\partial t}\frac{\partial U_L}{\partial\lambda}$$

$$+ \frac{K_L}{H^2}(1+\lambda^2(\partial H/\partial z)^2)\frac{\partial^2 U_L}{\partial\lambda^2} + \frac{K_L}{\lambda H^2}\frac{\partial U_L}{\partial\lambda} + K_L\frac{\partial^2 U_L}{\partial z^2} \qquad - (32)$$

$$- \frac{2K_L}{H}\lambda\frac{\partial H}{\partial z}\frac{\partial^2 U_L}{\partial\lambda\partial z} + [2\lambda(\partial H/\partial z)^2 - \lambda H\frac{\partial^2 H}{\partial z^2}]\frac{K_L}{H^2}\frac{\partial U_L}{\partial\lambda}$$

subject to the conditions

$$U_L(\lambda,z,0) = 1.0\ ,\quad 0 \leqslant \lambda < 1,\ 0 \leqslant z \leqslant D,\ t = 0.0 \qquad - (33)$$

$$U_L(\lambda,0,t) = 1.0\ ,\quad 0 \leqslant \lambda < 1,\ z = 0.0\ ,\ t > 0.0 \qquad - (34)$$

$$U_L(1,z,t) = 0.0\ ,\quad \lambda = 1.0\ ,\ 0 \leqslant z \leqslant D,\ t > 0.0 \qquad - (35)$$

$$\frac{\partial U_L}{\partial \lambda}(0,z,t) = 0.0\ ,\quad \lambda = 0.0\ ,\ 0 \leqslant z \leqslant D,\ t > 0.0 \qquad - (36)$$

For the interface we have

$$L\rho_S \frac{\partial H}{\partial t} = \frac{k_S(T_w-T_f)}{H(\ln R-\ln H)} - \frac{k_L}{H}(T_o-T_f)\frac{\partial U_L}{\partial \lambda} + h(T_f-T_w) \qquad - (37)$$

for $\lambda = 1.0$, $0 < z \leqslant D$ and $t > 0.0$, and subject to

$$H(z,0) = R(z)\ ,\quad 0 \leqslant z \leqslant D,\ t = 0.0 \qquad - (38)$$

$$H(0,t) = R_o\ ,\quad z = 0.0\ ,\ t \geqslant 0.0 \qquad - (39)$$

The new rectangular solution region is more suited for the application of finite difference schemes.

As the problem stands we would normally require to construct an iterative procedure to deal with the coupled nature of (32) and (37). However, further investigation provides a method for solving the equations almost independently. We note that the dependence of the interface location on the temperature is solely in terms of the 'radial' temperature gradient at the interface. Thus, a formula for evaluating this gradient without resorting to calculating the whole temperature distribution would effectively uncouple the interface from the temperature and reduce the computational effort required (the reverse is not true). In what follows such an expression is derived.

In the vicinity of the interface we neglect time dependence, radial velocity flow and axial conduction. In the original co-ordinate system the axial velocity is given by equation (7)

$$v_z = \frac{2Q}{H^2}\,[1-(r/H)^2](1-\tfrac{1}{2}\sin(\pi\omega t))$$

The energy equation (13) becomes

$$v_z \frac{\partial T_L}{\partial z} = \kappa_L \frac{\partial^2 T_L}{\partial r^2}\,, \quad 0 \leqslant r < H(z,t),\ 0 < z \leqslant D,\ t > 0.0$$

subject to

$$T_L(r,z,0) = T_o\,, \quad 0 \leqslant r < R(z),\ 0 \leqslant z \leqslant D,\ t = 0.0$$

$$T_L(r,0,t) = T_o\,, \quad 0 \leqslant r < R_o\,,\ z = 0.0\,,\ t > 0.0$$

$$T_L(H,z,t) = T_f\,, \quad r = H(z,t)\,,\ 0 \leqslant z \leqslant D,\ t > 0.0$$

We also suppose that $T_L(r,z,t) \to T_o$ as $r \to 0.0$ (which is reasonable for small κ_L). Using the normalisation $U_L = (T_L-T_f)/(T_o-T_f)$ we obtain

$$v_z \frac{\partial U_L}{\partial z} = \kappa_L \frac{\partial^2 U_L}{\partial r^2}\,, \quad 0 \leqslant r < H(z,t),\ 0 \leqslant z \leqslant D,\ t > 0.0$$

$$U_L(r,z,0) = 1.0\,, \quad 0 \leqslant r < R(z),\ 0 \leqslant z \leqslant D,\ t = 0.0$$

$$U_L(r,0,t) = 1.0\,, \quad 0 \leqslant r < R_o\,,\ z = 0.0\,,\ t > 0.0$$

$$U_L(H,z,t) = 0.0\,, \quad r = H(z,t)\,,\ 0 \leqslant z \leqslant D,\ t > 0.0$$

$$U_L(r,z,t) \to 1.0\,, \quad r \to 0.0\,,\ 0 < z \leqslant D,\ t > 0.0$$

Using the immobilisation transformation $\lambda = r/H(z,t)$ we have, assuming $\partial H/\partial z$ to be negligible,

$$v_z \frac{\partial U_L}{\partial z} = \frac{\kappa_L}{H^2} \frac{\partial^2 U_L}{\partial \lambda^2}\,, \quad 0 \leqslant \lambda < 1,\ 0 < z \leqslant D,\ t > 0.0$$

$$U_L(\lambda,z,0) = 1.0\,, \quad 0 \leqslant \lambda < 1,\ 0 \leqslant z \leqslant D,\ t = 0.0$$

$$U_L(\lambda,0,t) = 1.0\ ,\quad 0 \leqslant \lambda < 1,\ z = 0.0\ ,\ t > 0.0$$

$$U_L(1,z,t) = 0.0\ ,\quad \lambda = 1\ ,\ 0 \leqslant z \leqslant D,\ t > 0.0$$

$$U_L(\lambda,z,t) \to 1.0\ ,\quad \lambda \to 0\ ,\ 0 \leqslant z \leqslant D,\ t > 0.0$$

where

$$v_z = \frac{2Q}{H^2}(1-\lambda^2)\left(1-\frac{1}{2}\sin(\pi\omega t)\right)$$

The radial (λ) gradient of v_z at $\lambda = 1.0$ is

$$\frac{-4Q\,(1-0.5*\sin(\pi\omega t))}{H^2} \qquad -(40)$$

Near $\lambda = 1.0$ (i.e. near the interface) we assume that v_z may be represented by $\bar{v}_z$ which is linear in λ and has a gradient given by (40). If $\bar{v}_z = A(1-\lambda)$ then by differentiating and equating the result at $\lambda = 1.0$ to (40) yields

$$A = \frac{4Q}{H^2}\left(1-\frac{1}{2}\sin(\pi\omega t)\right)$$

and hence

$$\bar{v}_z = \frac{4Q\,(1-0.5*\sin(\pi\omega t))}{H^2}(1-\lambda)$$

Since all change in U_L takes place near $\lambda = 1.0$ we consider this a semi infinite problem in λ and obtain with $\xi = 1 - \lambda$

$$\bar{v}_z \frac{\partial U_L}{\partial z} = \frac{\kappa_L}{H^2}\frac{\partial^2 U}{\partial \xi^2}\ ,\quad \xi > 0.0,\ 0 < z \leqslant D,\ t > 0.0$$

$$U_L(\xi,z,0) = 1.0\ ,\quad \xi > 0.0,\ 0 \leqslant z \leqslant D,\ t = 0.0$$

$$U_L(\xi,0,t) = 1.0\ ,\quad \xi > 0.0,\ z = 0.0\ ,\ t > 0.0$$

$$U_L(0,z,t) = 0.0\ ,\quad \xi = 0.0,\ 0 \leqslant z \leqslant D,\ t > 0.0$$

$$U_L(\xi,z,t) \to 1.0\ ,\quad \xi \to \infty\ ,\ 0 \leqslant z \leqslant D,\ t > 0.0$$

$$\bar{v}_z = \frac{4Q\,\xi\,[1-0.5*\sin(\pi\omega t)]}{H^2}$$

Defining $\eta = \xi\,[4Q(1-0.5*\sin(\pi\omega t))/9K_L z]^{\frac{1}{3}}$, $\theta = 1-U_L$ and following Howarth (1953), p.773, we obtain

$$\frac{d^2\theta}{d\eta^2} + 3\eta^2 \frac{d\theta}{d\eta} = 0.0 \qquad - (41)$$

$$\theta = 1.0 \,,\ \eta = 0.0$$

$$\theta \to 0.0 \,,\ \eta \to \infty$$

The solution of (41) is $\theta(\eta) = \beta + K \int_0^{\eta} e^{-x^3} dx$ which, on using the boundary conditions, yields

$$\theta(\eta) = 1.0 - 0.120 \int_0^{\eta} e^{-x^3} dx$$

This equivalent to writing

$$\theta(\xi,z,t) = 1.0 - 0.120 \int_0^{\xi B} e^{-x^3} dx$$

where $B = [4Q(1-0.5*\sin(\pi\omega t))/9K_L z]^{\frac{1}{3}}$. At the interface ($\xi = 0.0$) the gradient is given by

$$\left.\frac{\partial U_L}{\partial \xi}\right|_{\xi=0} = -\left.\frac{\partial \theta}{\partial \xi}\right|_{\xi=0} = 0.12\left[\frac{4Q(1-0.5*\sin(\pi\omega t))}{9K_L z}\right]^{\frac{1}{3}}$$

Consequently,

$$\left.\frac{\partial U_L}{\partial \lambda}\right|_{\lambda=1} = -\left.\frac{\partial U_L}{\partial \xi}\right|_{\xi=0} = -0.12\left[\frac{4Q(1-0.5*\sin(\pi\omega t))}{9K_L z}\right]^{\frac{1}{3}} \qquad - (42)$$

With (42) as an approximation to the radial (λ) temperature gradient at the interface (λ=1), equation (37) may be solved without the use of

equation (32). This allows us to observe the behaviour of the interface with a view to modifying equation (32).

4.5 A Numerical Solution

We shall initially consider the explicit difference solution (see appendix 4.2 for the difference equation) of equation (37).

4.5.1 The Interface Location

We first require an estimate of the solid and liquid thermal diffusivities, K_S and K_L, which may be determined from $K = k/(\rho c)$. Given the specific heat c and density ρ we determine an average thermal conductivity k_{av} using equations (4c) and [Carslaw and Jaeger (1959), p.93]

$$k_{av} = \frac{1}{T_2 - T_1} \int_{T_1}^{T_2} k(T)\, dT$$

Assuming that the restrictions $T_w < 1000^{\circ}C$, $T_f > 1000^{\circ}C$ and $T_o > T_f$ hold then we obtain for the average solid and liquid thermal conductivities

$$k_{s.av} = 0.96089239\, [877.5 - 0.65T_w(1+3.5\times10^{-4}T_w) - 1.12T_f(1-10^{-3}T_f)]$$

$$k_{l.av} = 1.07619948\, [-1 + 10^{-3}(T_o+T_f)]$$

These values are used in all subsequent calculations. Secondly, from equation (42), it is clear that we require an estimate of the

volume flux Q(t) before evaluating the 'radial' temperature gradient. Q(t) is a function of the interface location. Over the length $0 \leqslant z \leqslant D$, given a pressure drop P, a kinematic viscosity ν and liquid density ρ_L the volume flux is given by

$$Q(t) = \frac{P}{8\rho_L \nu \int_0^D H^{-4} dz}$$

At t = 0.0, $H(z,t) = R(z) = R_o + z \cot(\alpha\pi)$ and the initial volume flux is

$$Q(0) = \frac{P}{8\rho_L \nu \int_0^D R^{-4} dz}$$

from which we obtain

$$Q(t) = Q(0) \frac{\int_0^D R^{-4} dz}{\int_0^D H^{-4} dz} \qquad - (43)$$

Assuming an initial inlet velocity (bulk) of v_o, and noting the inlet/outlet radius is R_o, we have

$$Q(0) = \pi R_o^2 v_o \qquad - (44)$$

By evaluating the integral $\int_0^D R^{-4} dz$ and substituting (44) in (43) we obtain

$$Q(t) = \frac{\bar{D}}{\int_0^D H^{-4} dz} \qquad - (45)$$

where the constant $\bar{D}$ is given by

$$\bar{D} = \pi v_o D \frac{[3R_o^2 + 3R_o D\cot(\alpha\pi) + D^2\cot^2(\alpha\pi)]}{3R_o [R_o + D\cot(\alpha\pi)]^3}$$

The integral $\int_0^D H^{-4} dz$ is evaluated at each time step using the latest available interface location. If the number of axial nodes is <u>even</u> a composite form of Simpson's Rule is used. If the number of axial nodes is <u>odd</u> we use Gregory's Formula which provides a similar accuracy to Simpson's Rule [Phillips and Taylor (1973), p.132].

We should also note that initially the interface velocity is very high (infinite at t = 0.0) and so to track the interface at consistent intervals of distance the initial time steps may need to be small. A final consideration is the frequency of oscillation of the liquid. To ensure that the oscillatory behaviour is incorporated into the model to a reasonable extent we use a minimum of 10 time steps per oscillation, one oscillation being defined as the time between the velocity pulsing. This effect is not considered if $\omega = 0.0$ (steady <u>uni-directional</u> flow) or $v_o = 0.0$ (a <u>stagnent</u> problem). Thus the time step is determined as being the minimum of

1. Time taken for the interface at $z = \Delta z$ to move 0.05mm.
2. The time $1/10\omega$ (if $\omega > 0.0$).
3. The time 0.4s.

To obtain a starting solution we utilize the one-dimensional solution. The freezing parameter μ is first calculated using equation (4). The error function, erf(x), is calculated by evaluating the integral representation using Simpson's Rule with 100 points. This yields an accuracy $|\text{erf}(x)_{ex} - \text{erf}(x)_{num}| < 10^{-8}$ for an argument $0 \leqslant |x| \leqslant 3/2$. From $x_o = 2\mu(\kappa_s t_o)^{\frac{1}{2}}$ we obtain the approximate time taken for the interface to move the first 0.05mm as $t_o \simeq 6.25\text{x}10^{-10}/(\mu^2 \kappa_s)$. We can then estimate the volume flux $Q(t_o)$ from equation (45) and the temperature gradient from equation (42).

4.5.2 Results and Discussion

Several simulations were completed (for a listing of the code see appendix 4.3) to test the dependence of the interface location with regard to several relevant physical quantities (figures 4.10 to 4.18). Since the interface calculation is independent of the temperature distribution the above results were obtained without computing the temperature and hence a larger maximum time step was practical. Due to the non-linearity of the interface equation (37) the stability of the numerical scheme is difficult to evaluate but calculations using a maximum time step of 0.4s seemed to cause no adverse effects.

The initial conditions were estimated as described in section 4.5.1 and the program was then used to simulate the conditions existing at a time of 15 minutes. The following nine parameters were varied independently to judge their effect on the resulting crust thickness - wall temperature (T_w), fusion temperature (T_f), initial liquid temperature (T_o), latent heat of fusion (L), convective heat

transfer coefficient (h), initial bulk velocity (v_o), bubble frequency (ω), inlet radius (R_o) and wall elevation ($\alpha\pi$ radians). For each simulation the values of the parameters not being tested are shown in table 4.4.

Table 4.4

Standard Parameter Values

Wall Temperature	:	100.0	^oC
Fusion Temperature	:	1300.0	^oC
Initial Liquid Temperature	:	1400.0	^oC
Latent Heat of Fusion	:	216000.0	J/kg
Convective Heat Coefficient	:	200.0	J/m s ^oC
Initial Slag Velocity	:	0.05	m/s
Bubble Frequency	:	0.0	s^{-1}
Inlet Radius	:	0.25	m
Wall Elevation	:	90.0	degrees

In addition, table 4.5 gives the various constant values used in the calculations.

Table 4.5

Constant Data

Liquid Density	:	2247.5654	kg/m^3
Solid Density	:	2562.9546	kg/m^3
Liquid Specific Heat	:	1179.5520	J/kg^oC
Solid Specific Heat	:	1127.1558	J/kg^oC
Tube Length (slag depth)	:	0.5	m

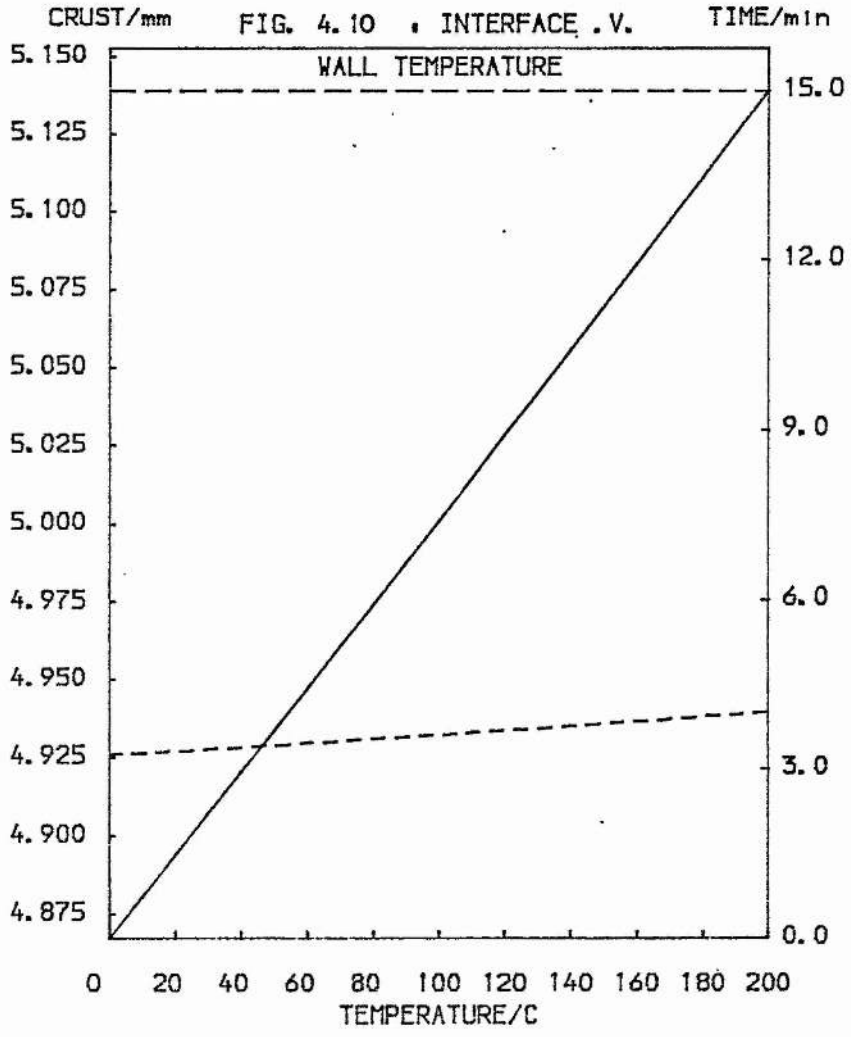


FIG. 4.10 : INTERFACE .V.

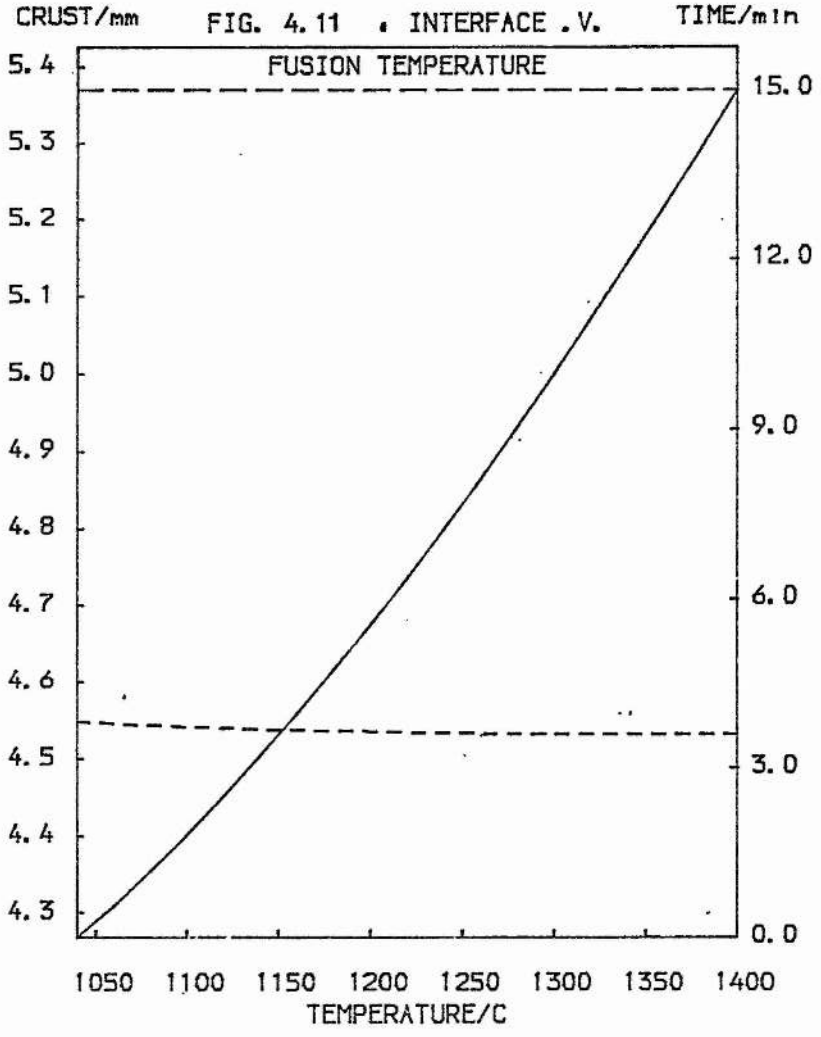


FIG. 4.11 : INTERFACE .V.

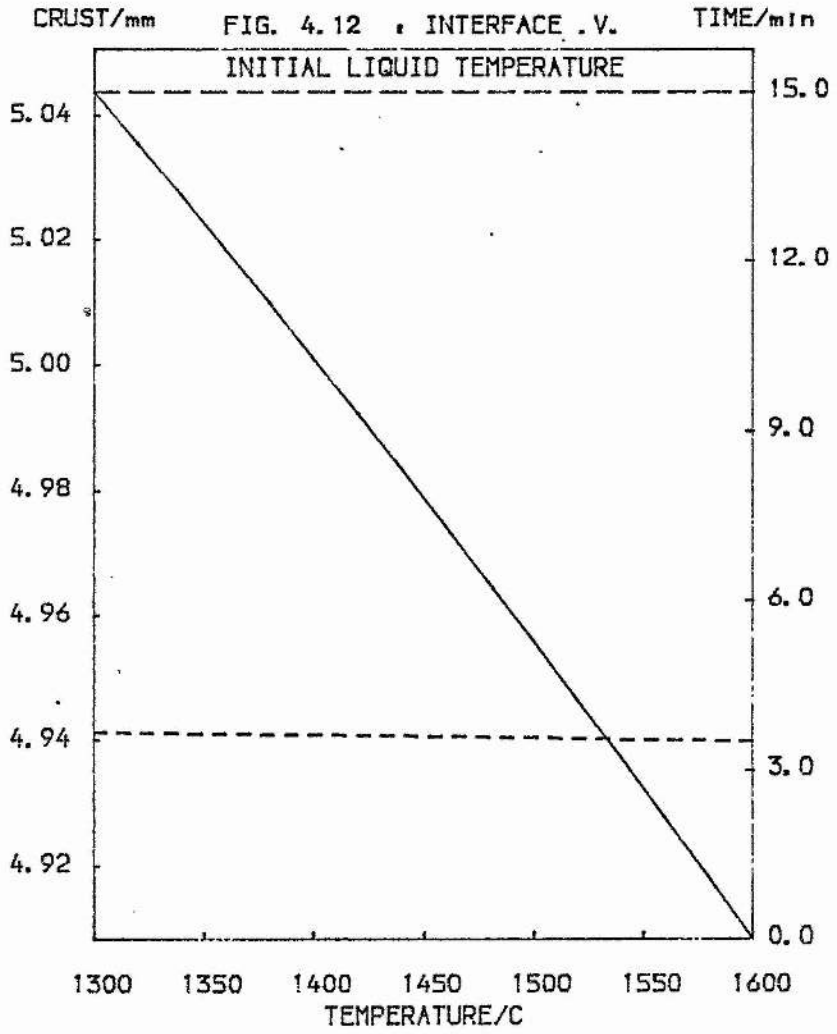


FIG. 4.12 : INTERFACE . V.

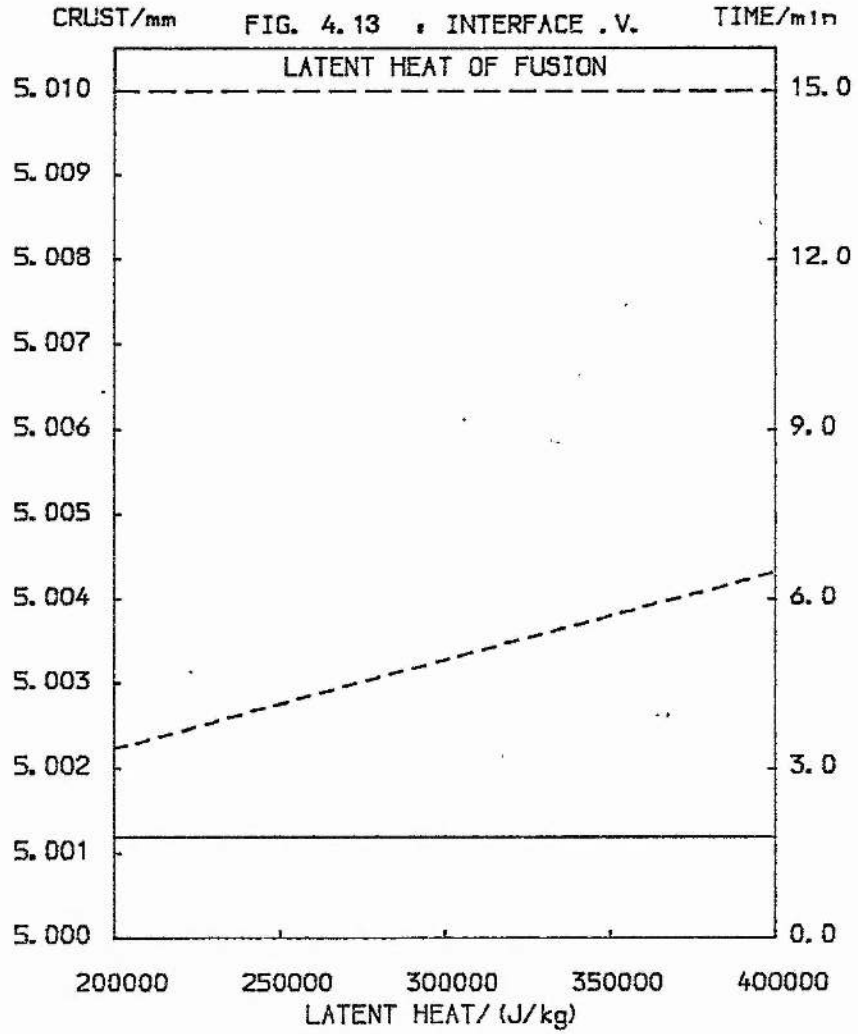
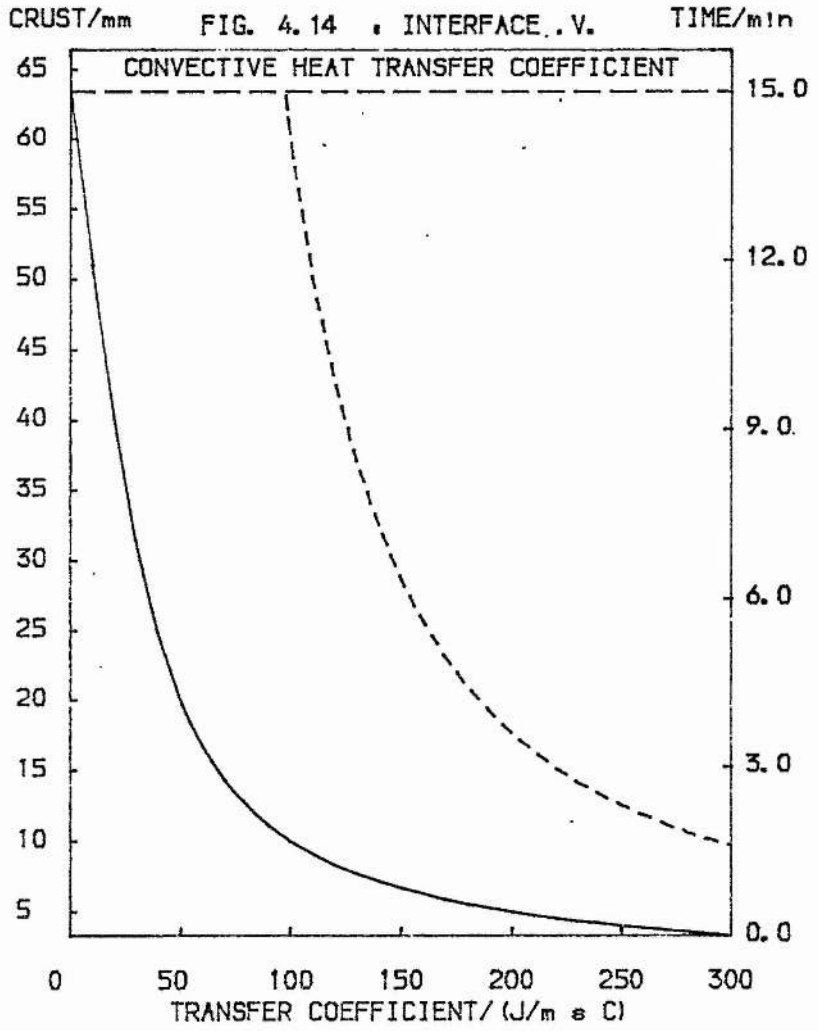
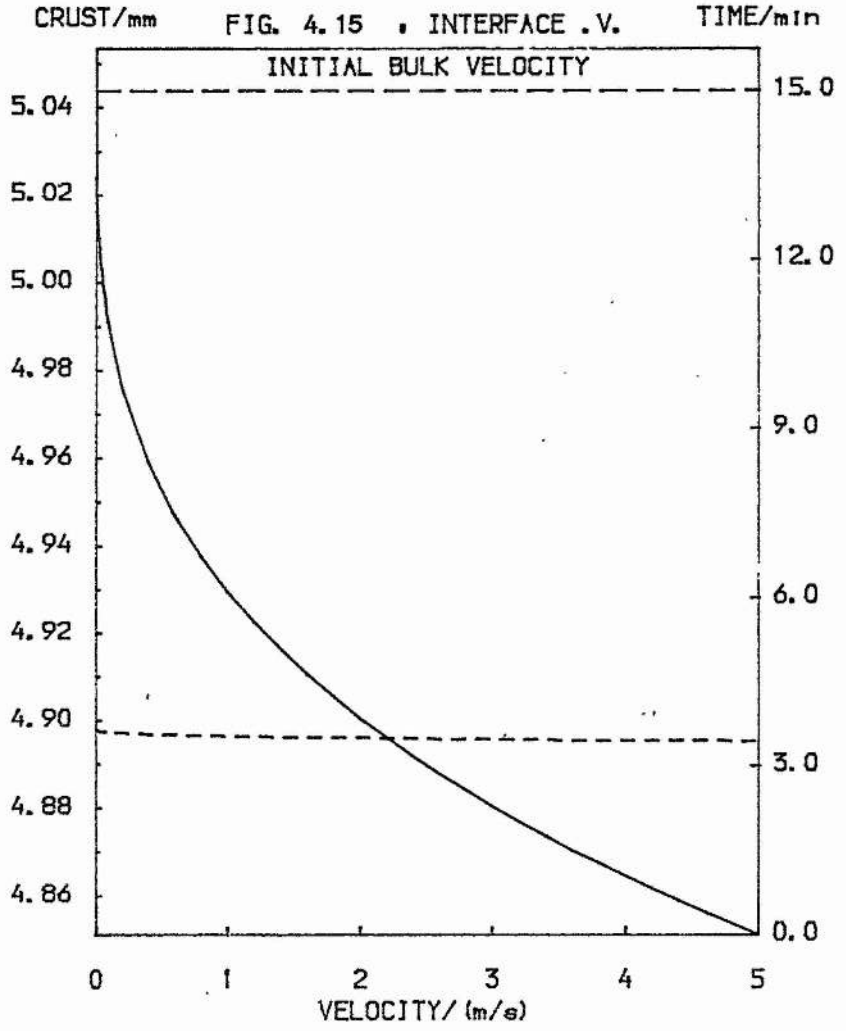


FIG. 4.13 : INTERFACE . V.


CRUST/mm
FIG. 4.14 . INTERFACE . V.
TIME/min
CONVECTIVE HEAT TRANSFER COEFFICIENT
65
60
55
50
45
40
35
30
25
20
15
10
5
15.0
12.0
9.0
6.0
3.0
0.0
0
50
100
150
200
250
300
TRANSFER COEFFICIENT/ (J/m s C)


CRUST/mm
FIG. 4.15 . INTERFACE . V.
TIME/min
INITIAL BULK VELOCITY
5.04
5.02
5.00
4.98
4.96
4.94
4.92
4.90
4.88
4.86
15.0
12.0
9.0
6.0
3.0
0.0
0
1
2
3
4
5
VELOCITY/ (m/s)

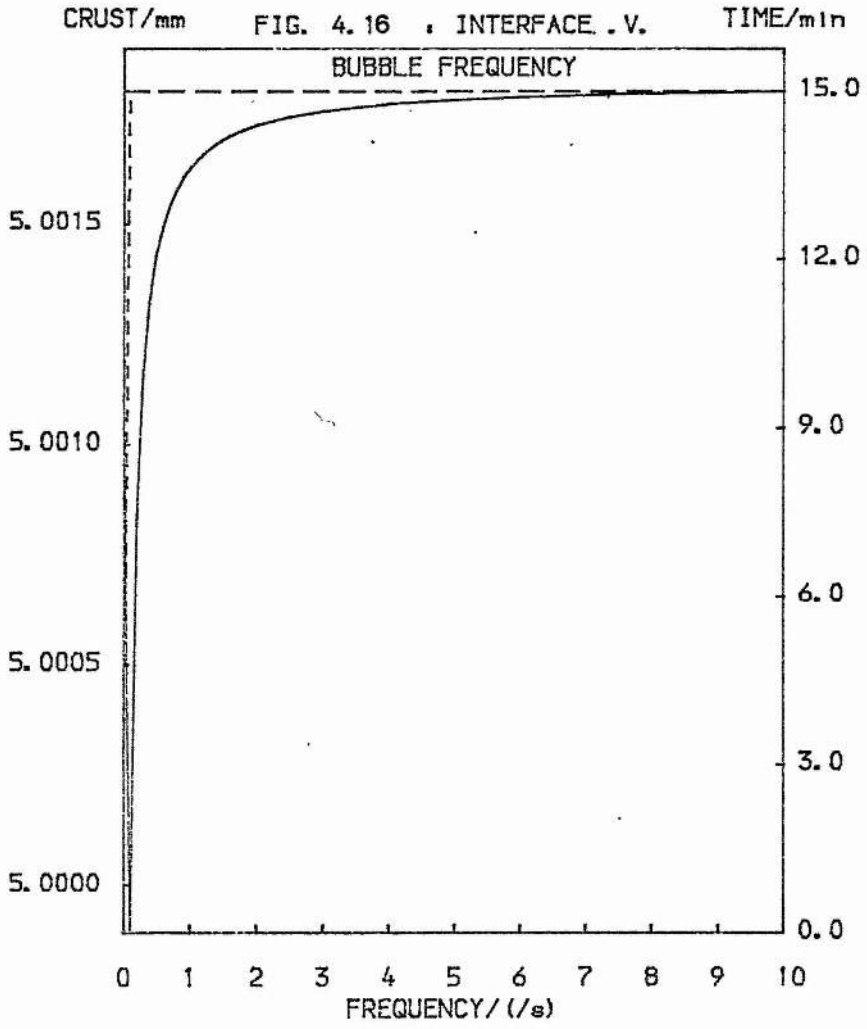


FIG. 4.16 : INTERFACE . V.

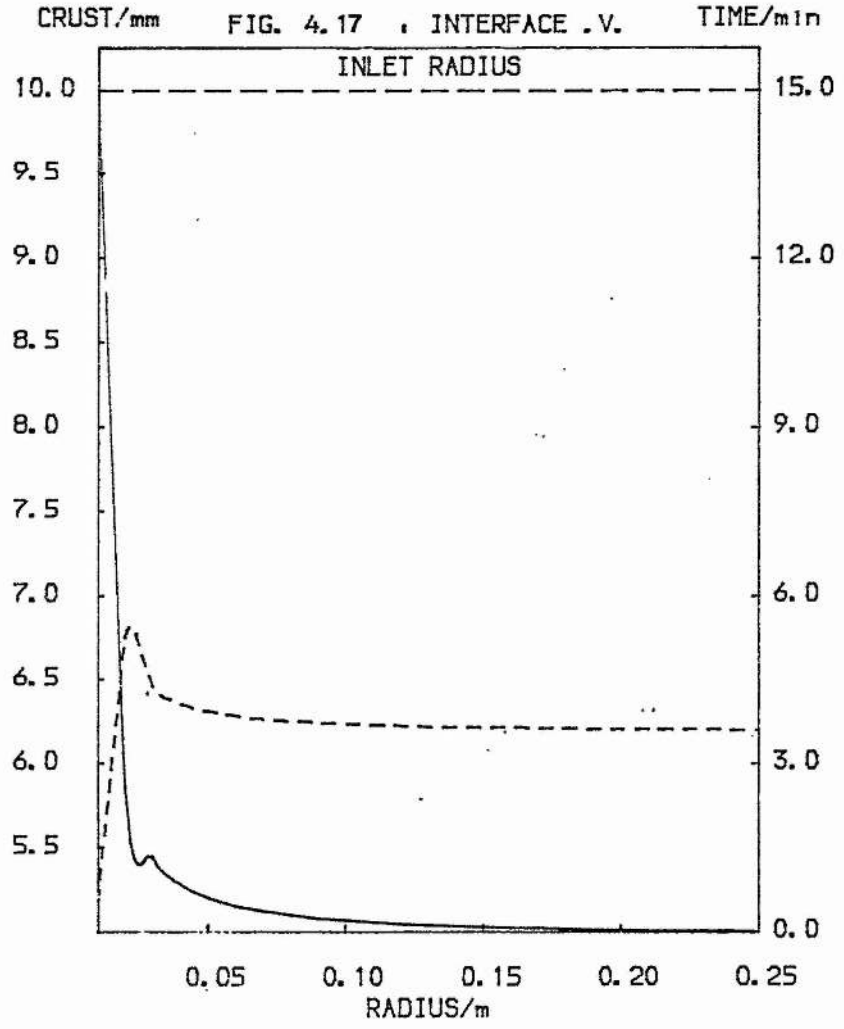
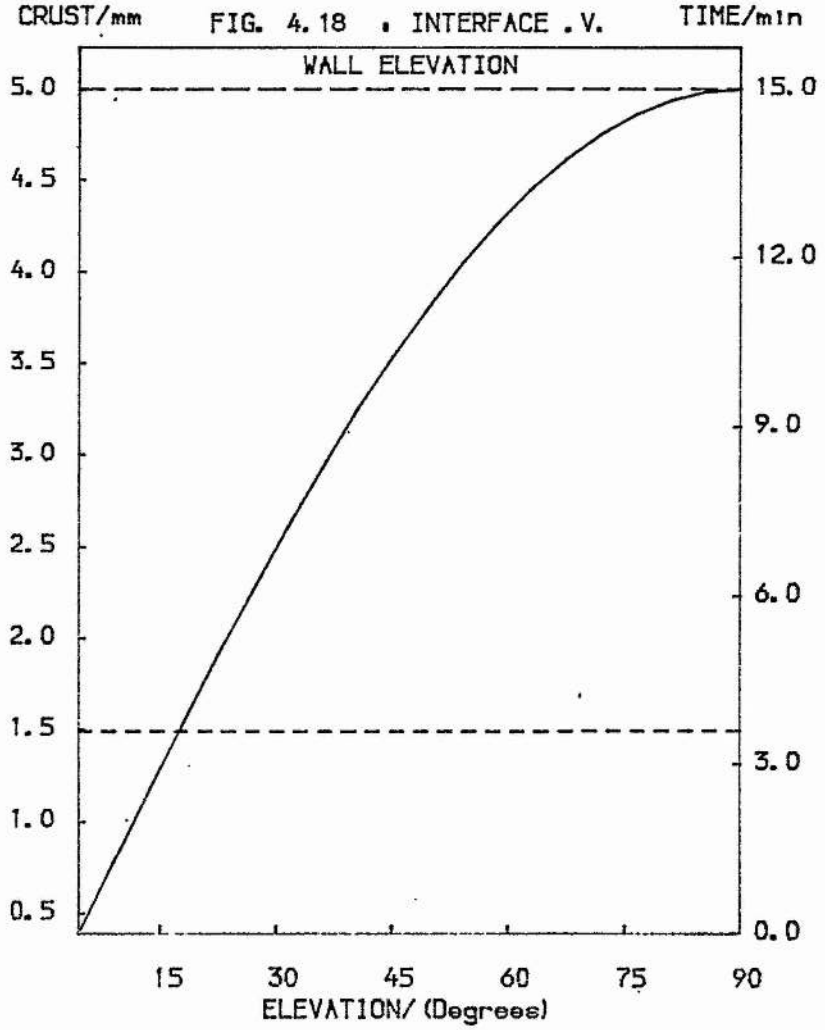


FIG. 4.17 : INTERFACE . V.


CRUST/mm
FIG. 4.18 . INTERFACE . V.
TIME/min
WALL ELEVATION
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
15.0
12.0
9.0
6.0
3.0
0.0
15
30
45
60
75
90
ELEVATION/ (Degrees)

In all cases the thickness of the crust is calculated at a distance of 0.25m from the inlet, that is, half way along the tube. This thickness is closely representative of the crust thickness at points away from the vicinity of the inlet, as seen in figure 4.19a which shows the progressive interface profile for the parameter values given in table 4.4.

Figures 4.10 to 4.18 show the steady-state crust thickness (solid line), if such a situation exists, as a function of one of the afore-mentioned parameters. The broken curve shows the real time taken to reach a steady-state situation for times less than t_{max} (15 minutes) as shown by the straight broken line across the top of each figure. Values of t = 15 minutes imply that a steady-state condition has not yet been reached.

The first observation that we should note is that the existence of a steady-state solution would appear to be critically dependent on the convective heat transfer coefficient. From figure 4.14 (for the values of table 4.4) it is seen that a coefficient of approximately 100 $J/ms^{o}C$ is required. For values below 100 $J/ms^{o}C$ the crust thickness (at 15 minutes) is very significantly dependent on h. As h increases from 0.0 to 100.0 so the dependence varies from $-9.491x10^{-1}$ $mm/(J/ms^{o}C)$ to $-1.402x10^{-1}$ $mm/(J/ms^{o}C)$. For values of 100 $J/ms^{o}C$ and greater the sensitivity of the steady-state crust thickness rapidly reduces (to $-1.15x10^{-2}$ $mm/(J/ms^{o}C)$ at h = 300 $J/ms^{o}C$).

With regard to the bubble frequency (figure 4.16) it would appear that any unsteady (oscillatory) motion excludes the possibility of a steady-state solution (at least for t = 15 minutes). The curve from $\omega = 0.0\ s^{-1}$ to $\omega = 0.1\ s^{-1}$ is a bit misleading. These are the first

two values of ω considered for this figure. The first ($\omega = 0.0s^{-1}$) produces a steady state thickness of 5.0012 mm at t = 215.62 s. The value $\omega = 0.1s^{-1}$ produces no steady-state solution and thus the curve between these two points is (possibly) rather meaningless. However, the requirement of a steady uni-directional velocity for a steady-state condition would appear to exist. For non-zero values of the crust thickness at 15 minutes varies very little with ω (maximum: 4.9999 mm, minimum: 5.0018 mm). These values are very close (maximum deviation is ~0.012%) to the steady-state thickness for $\omega = 0.0s^{-1}$. In fact, as ω increases it appears that this 'non-steady-state' crust thickness is tending towards an asymtotic limit of approximately 5.0018 mm. Noting the steady-state time, t_s, we see that the fastest average interface velocity for $\omega > 0.0s^{-1}$ to 900.0s (15 minutes) is 5.555×10^{-3} mm/s whereas up to t_s (for $\omega = 0.0s^{-1}$) the average velocity is 23.195 mm/s. Thus, even for simulation times of greater than 15 minutes it is unlikely that a non-zero ω would be the cause of a blockage situation.

Figure 4.15 shows the steady-state crust thickness as a function of the initial bulk velocity v_o. The existence of a steady-state condition does not seem to be dependent on velocity although it is clear that for lower velocities (from 0.0 m/s to 0.5 m/s) the steady-state thickness is considerably affected by changes in v_o, the gradient of the curve decreasing from -3.989×10^{-1} mm/(m/s) to -2.833×10^{-1} mm/(m/s). By the time v_o reaches 5.0 m/s the gradient has reduced to -0.012 mm/(m/s). For all values of v_o the time taken to reach a steady-state condition is relatively constant, ranging from 218.55s to 205.3s as v_o increases from 0.0 m/s to 5.0 m/s.

With regard to the inlet radius R_o (figure 4.17) the crust

thickness is acutely dependent for values of $R_o < 0.025$ m. For the first point on the curve (R_o = 10 mm) blockage occurs (at t = 32.98s) which accounts for the 'hump' in the time curve (broken line). For $R_o > 0.02$ m a steady-state condition exists with the steady-state time trailing off rapidly from 317s to a fairly constant value of between 221s and 215.6s as R_o increases from 0.1 m to 0.25 m.

The slag pool wall elevation (figure 4.18) has virtually no effect on the time taken to reach steady-state conditions. The time ranges from 214.34s to 215.62s. Starting from a value of 4.5 degrees ($\pi/40$ radians) the crust thickness normal to a point half way along the wall increases from 0.39 mm to 5.00 mm at 90.0 degrees (a parallel cylindrical tube). This is possibly explained by the fact that as the elevation is increased so the effective tube radius is being decreased and from figure 4.17 we saw that a decreasing inlet radius (for a parallel tube) increased the crust thickness. However, the dependence is quite different with the gradient decreasing from 8.597×10^{-2} mm/degree (at 4.5 degrees) to 3.583×10^{-3} mm/degree (at 90.0 degrees). That is, the dependence decreases as the effective radius is reduced.

The latent heat L (figure 4.13) has absolutely no effect on the steady-state crust thickness which is expected since L is really just a measure the of the interface velocity. As such, we see from the diagram that as L increases from 2×10^5 J/kg to 4×10^5 J/kg the time taken to reach a steady-state condition increases linearly from 200.2s to 389.24s. Thus, the average speed of the interface decreases (from 2.498×10^{-2} mm/s to 1.285×10^{-2} mm/s) which is in agreement with the one-dimensional results of figure 4.7 (3.277×10^{-2} mm/s to 2.924×10^{-2} mm/s at 900.0s).

The dependence of the steady-state crust thickness upon the temperatures T_w, T_f and T_o are displayed in figures 4.10, 4.11 and 4.12 respectively. The existence of steady-state conditions does not seem to be dependent on any of them. With respect to the wall temperature, T_w, the steady-state dependence is almost linear at 1.359×10^{-3} mm/^{o}C. The average velocities at $T_w = 100^{o}$C and $T_w = 200^{o}$C are 2.3193×10^{-2} mm/s and 2.1329×10^{-2} mm/s respectively. These correspond to steady-state times of 216s and 240s. Thus, the average dependence of crust thickness at constant time (228s, as an average time) is approximately -4.26×10^{-3} mm/^{o}C which compares favourably with the one dimensional problem (figure 4.4). Similarly, the steady-state dependence on the fusion temperature T_f ranges from 2.0660×10^{-3} mm/^{o}C to 3.0439×10^{-3} mm/^{o}C with the greater sensitivity at values of T_f close to T_o (as evidenced in section 4.3). The average dependence of the crust at constant time (222s) is approximately 3.81×10^{-3} mm/^{o}C which is an order of magnitude less than the one dimensional values (figure 4.3). This is most probably due to the considerable effect that the convective heat coefficient has on the solution. With a value of h = 0.0 J/m s^{o}C a similar calculation to that above yields a (constant time) dependence of 6.57×10^{-2} mm/^{o}C which compares more favourably with figure 4.3. Finally, repeating the analysis for the initial liquid temperature, T_o, the steady-state crust dependence ranges from -4.1708×10^{-4} mm/^{o}C to -4.8444×10^{-4} mm/^{o}C. The average crust dependence at constant time (214s) is 2.599×10^{-4} mm/^{o}C which is two orders of magnitude less than the one dimensional results (figure 4.2). Repeating the above simulation with h = 0.0 J/m s^{o}C realises an average (constant time) crust dependence of -2.26×10^{-2} mm/^{o}C which, again, compares well with

figure 4.2. So, in conclusion, we see that the steady-state crust thickness is not particularly sensitive to variations in any of the three mentioned temperatures (although we might have expected this with T_w) and this would appear to be because of the domination of h on the solution.

Figures 4.19a & b and 4.20a & b show the progressive interface profile shape and motion of one point on the interface as the situations of steady-state and blockage are approached. From figure 4.19a it is evident that the crust reaches a fairly uniform thickness in a small distance from the inlet. This aspect is more evident in figure 4.20a. With regard to a steady-state solution we see (figure 4.19b) that the majority of the crust formation takes place in (approximately) the first third of the total time taken. This is also seen in figure 4.19a where we see the profiles becoming closer together at successive (constant) intervals of time.

The results are quite different for the situation approaching blockage. Figure 4.20b, for a point half way along the tube, shows the interface velocity to be initially 'fast' and slowing down, though not as dramatically as in figure 4.19b. However, once the half way time (approximately) has been passed the velocity increases as shown by the 'S' type curve. This trend continues until blockage (at 20mm).

4.5.3 The Temperature Distribution

From the evidence presented in section 4.5.2 it is clear that the value $\partial H/\partial z$ is very small away from the vicinity of the inlet/outlet ($z = 0.0$). The value of $\partial^2 H/\partial z^2$ will be smaller. Thus, in equation (32) we shall neglect all terms containing these two values. Also

FIG. 4.19a . INTERFACE APPROACHING STEADY-STATE

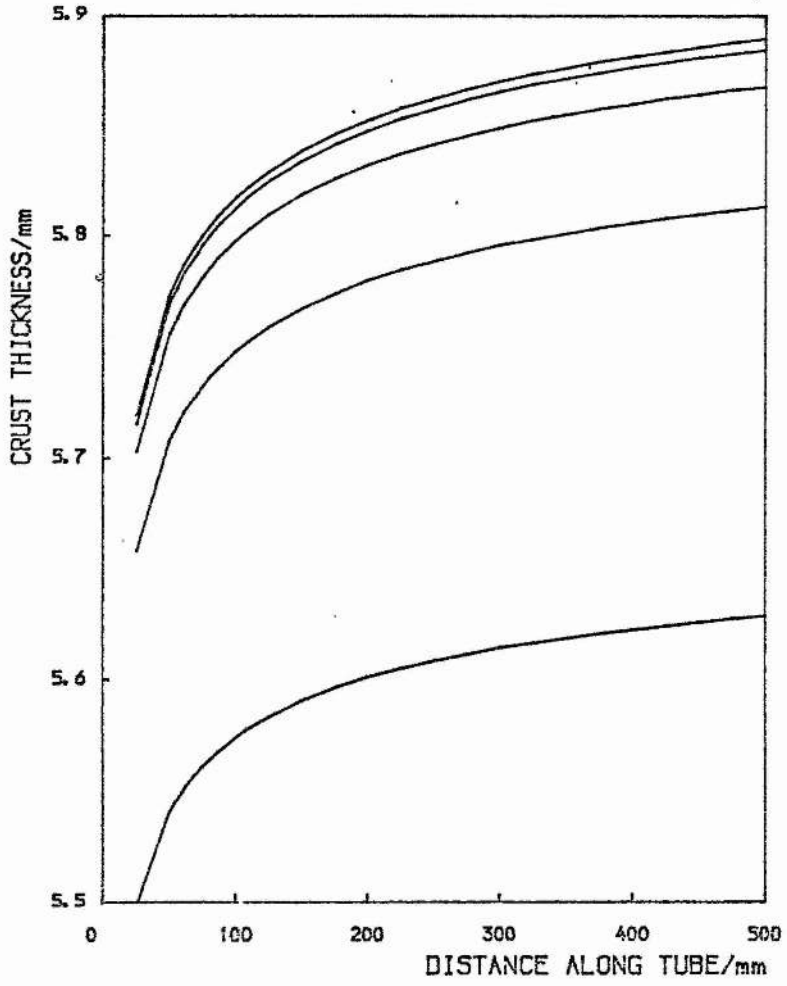


FIG. 4.19b . CRUST THICKNESS APPROACHING STEADY-STATE

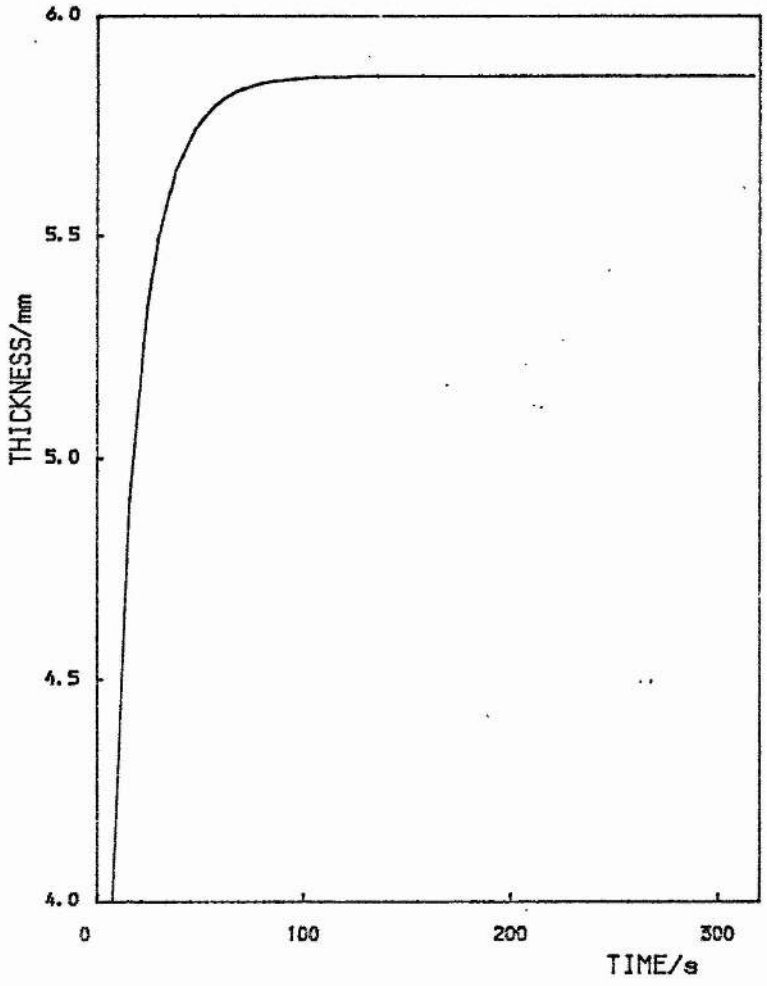


FIG. 4.20a : INTERFACE APPROACHING BLOCKAGE

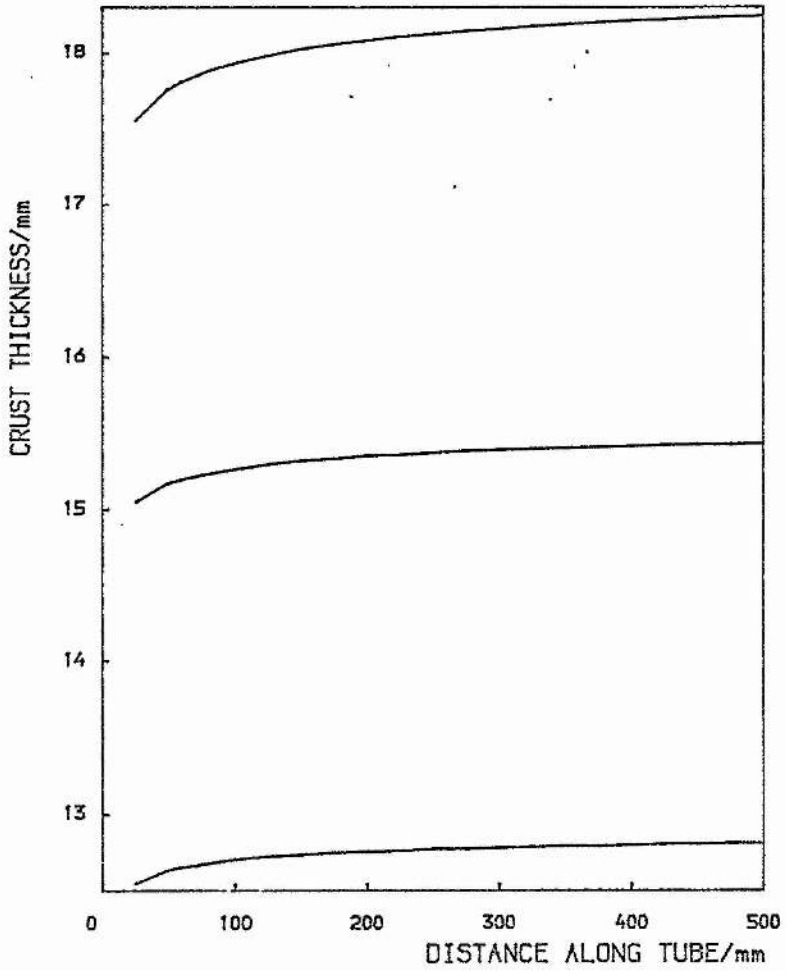
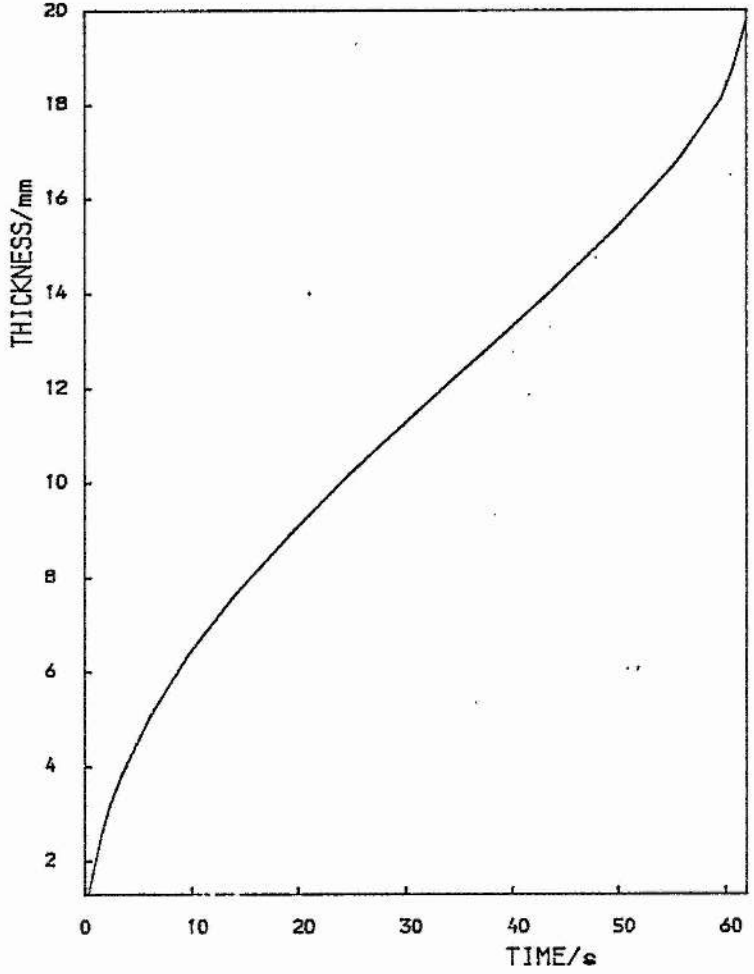


FIG. 4.20b : CRUST THICKNESS APPROACHING BLOCKAGE



because of the low values of the interface velocity $\partial H/\partial t$ we shall assume that the liquid velocity and temperature distributions are going to adapt to changes in the interface as quickly as they occur. As such the terms $\partial U_L/\partial t$ and $\frac{\lambda}{H}\frac{\partial H}{\partial t}\frac{\partial U}{\partial \lambda}$ will be neglected. Equation (32) reduces to

$$2Q(1-\lambda^2)\ (1 - \frac{\sin(\pi\omega t)}{2})\ \frac{\partial U}{\partial z} = K_L(\frac{\partial^2 U}{\partial \lambda^2} + \frac{1}{\lambda}\frac{\partial U}{\partial \lambda}) \qquad - (46)$$

$$0 \leqslant \lambda < 1,\ 0 < z \leqslant D,\ t > 0$$

subject to the conditions (33) - (36). Due to the presence of the volume flux Q(t) we will also require to re-solve equation (37). Preliminary calculations would appear to imply an upper limit of approximately 0.1s on the time-step. Thus, the time-step is calculated as in section 4.5.1 with this as the third condition.

We continue as follows. Initially a rectangular grid is constructed over the solution region $0 \leqslant \lambda \leqslant 1$, $0 \leqslant z \leqslant D$. Given N_z axial segments, the axial mesh size is given by $z = D/N_z$. For the radial mesh size we note that most of the temperature change takes place in the vicinity of the interface. Accordingly we use a graduated radial mesh length which decreases as λ increases. Given N radial segments, the i^{th} mesh length is given by $\Delta\lambda_i = k(1-\lambda_i)^{\frac{1}{2}}$ with the condition $\sum_{i=1}^{N_\lambda} \Delta\lambda_i = 1.0$ which determines k. Infact, it is found that $\Delta\lambda_i = a_i k^2$ where

$$a_1 = 1.0\ ,\quad a_i = \frac{1}{2}\left[1 + (1 + 4\sum_{j=1}^{i-1} a_j)\right],\ i = 2,3,\ldots,N$$

k is now determined from $k^2 \sum_{i=1}^{N_\lambda} a_i = 1.0$.

To solve equation (46) we adopt the following procedure. First we difference the right-hand side of (46) (see appendix 4.2). Assuming that the temperature is changing slowly with time the difference approximation of (46) may be written in the form

$$\frac{\partial U_L}{\partial z} = f(z,t,U_L)$$

which is in a suitable form to implement a standard one-step numerical integration procedure. In this case we use the classical Runga-Kutta method [see, for example, Phillips and Taylor (1973), p.292] which takes the form

$$U_{i,j} = U_{i,j-1} + \{dz(k1 + 2(k2 + k3) + k4)\}/6$$

A simple backward difference form is used for the z-derivitive. The k_i represent f(z,t,U) evaluated at four points. Thus the overall basic solution scheme may be described as :

1. Calculate the volume flux Q(t) (subroutine VOLUFLUX)
2. Calculate $\frac{\partial U_L}{\partial \lambda}$ at $\lambda = 1.0$ using (42).
3. Calculate $\frac{\partial H}{\partial t}$, the interface velocity, using (37).
4. Calculate the time-step dt
5. Calculate the new interface location from $H_j^{m+1} = H_j^m + \Delta t \frac{\partial H}{\partial t}$.
6. Test for a blockage or steady-state condition.
7. Calculate the new temperature profile from (46) using the Runga-Kutta scheme outlined above (also see appendix 4.2)

The routine PROGKUTT that performs calculations 2 to 7 is listed

at the end of appendix 4.3. It takes the place of the routine PROGFACE which is used for computing the interface solution alone.

The program was first run using the data shown in table 4.4 and table 4.5, that is, with no pulsating motion of the liquid. This produced a steady-state (interface) solution in 218.16s (with a maximum time step of 0.1s) which compares well with that obtained in section 4.5.2 of 215.62s (with a maximum time step of 0.4s). The results indicate that the temperature has reached a steady-state situation (within the tolerance of the solution) in approximately 1/3 of the time for a steady-state interface. To three figures it is only the temperature at the nodes immeadiately adjacent to the interface that have changed (except for the nine points furthest downstream from the inlet). The values are shown in table 4.6(a) below.

Table 4.6(a)

Steady-state Temperatures at intervals of $2\Delta z$ for the last three radial nodes (the remaining values are unity)

1.000	1.000	1.000	1.000	1.000	1.000	0.999	0.999	0.999	0.999	0.999
1.000	0.993	0.986	0.979	0.972	0.965	0.958	0.952	0.945	0.939	0.932
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Introducing a pulsating motion with a bubble frequency of $1.0s^{-1}$ produces even less change in the temperature distribution as seen in table 4.6(b) below.

Table 4.6(b)

Steady-state temperatures at intervals of $2\Delta z$ adjacent to the interface with $\omega = 1.0$ /s (remaining values are unity)

1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.999	0.999	0.999
1.000	0.995	0.990	0.984	0.979	0.974	0.969	0.964	0.959	0.955	0.950
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

The temperature change is restricted to the nodes adjacent to the interface at all axial locations (except for the last five nodes). This situation was reached within 18s. For this data no steady-state interface is reached by the maximum time of 15 minutes and so the 18s represents 2% of the total simulation time, that is, all temperature change is completed very quickly.

If we consider a situation in which there is a significantly larger movement of the interface the result is slightly changed. Taking the convective heat co-efficient to be zero, which implies no steady-state situation with the values of the remaining parameters as they are (table 4.4), we see the temperature at 15 minutes has changed considerably more although this change is restricted to the two points adjacent to the interface (see table 4.6(c)).

Table 4.6(c)

Temperatures at 15 minutes at intervals of $2\Delta z$ for the last three radial nodes (the remaining values are unity)

1.000	1.000	0.999	0.999	0.998	0.997	0.996	0.994	0.993	0.991	0.989
1.000	0.979	0.959	0.940	0.922	0.904	0.887	0.871	0.855	0.840	0.826
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

The temperature changed consistently throughout the simulation.

As far as the temperature simulation is concerned it is not possible to obtain a solution approaching blockage. By definition a blockage situation implies a zero volume flux. This in turn produces a zero co-efficient for the term $\frac{\partial U_L}{\partial z}$ in the energy equation. The final outcome is a zero divisor in the Runga-Kutta algorithm. In practice the solution breaks down before blockage is reached due to an arithmetic overflow FORTRAN error. This problem does not affect the interface evaluation.

4.6 Conclusions

A very complex industrial freezing problem has been described and the potentially complicated solution has been reduced to the solution of two 'weakly' coupled partial differential equations governing the temperature distribution and interface location. It is evident from the results obtained that the temperature is virtually unaffected and that changes only occur in the immeadiate vicinity of the interface. This may be a consequence of the boundary condition at z = 0.0. Possibly a more realistic condition would be one of constant (or slowly varying) heat flux although determining a suitable value may be difficult.

However, the interface location would appear more sensitive to changes in the physical surroundings and we have indicated the effects that several of the physical parameters have on the solution. In particular we see that the crust thickness is largely dependent upon the values of the convective heat co-efficient, the radius of the inlet (for 'smaller' values) and the bulk velocity. It is primarily

these values which seem to determine the existence of steady-state or blockage situations.

Although a non-zero bubble frequency pre-empts a steady-state solution the value of the crust velocity is so reduced that the thickness after 15 minutes differs only marginally from the steady-state value obtained with $\omega = 0.0s^{-1}$. It is also observed that the value of this '15 minute' crust thickness is moving asymptotically towards a psuedo-steady-state value as ω increases which would indicate a reduction in the sensitivity of the crust with regard to the bubble frequency.

Another interesting observation is the fact that at any given time the interface attains a fairly uniform thickness in a very short distance from the inlet/outlet. This fact is emphasised in figures 4.19(a) and 4.20(a).

SUMMARY

In this thesis we have investigated a collection of Stefan problems, ranging from the classical one-dimensional problem to a three-dimensional industrial problem, from both the computational point of view and from the practical point of view of obtaining a solution to a 'real life' freezing problem.

In the first instance a computationally cheap and relatively accurate numerical scheme, consisting of the enthalpy method and hopscotch difference algorithm, was proposed and applied to the classical Stefan problem suitably normalised. The results compared well with a selection of existing solution techniques and with the analytic solution. In certain cases it is significant to note that considerable savings in CPU time were made.

To test this scheme in a more complicated situation a two-dimensional problem with a boundary singularity was formulated (which for good measure may be regarded as a crude model of some industrial heating applications). This model has no apparent analytic solution. However, a two-dimensional test problem was constructed which contains the essential characteristics of the original problem that make it a good test for numerical schemes. This test problem also has an analytic solution.

The numerical scheme provided a very good description of the overall temperature distribution, the effect of the singularity being very localised. A minor modification to take into account the local temperature behaviour about this point improved the solution.

The scheme was then applied to the original 'element' problem and with regard to the results of the test problem a very reasonable

solution was obtained.

In the course of this investigation an attempt to obtain the analytic solution to the 'element' problem was made. The approach was to solve the problem outside an ellipse and then allow the eccentricity of the bounding ellipse to tend to unity which 'reduces' the ellipse to an element. This work was partially successful in that a pseudo-analytic solution was obtained using Lightfoot's (1930) 'moving source' technique. However, such was the complexity of the resulting integro-differential equation that no numerical evaluation has as yet been possible.

As part of the research project a certain amount of collaboration with British Gas was required. Through this contact the extension to a three-dimensional industrial problem was undertaken with the requirement of obtaining a practical, working solution. Using several simplifying assumptions the problem was reduced to a two-dimensional freezing model concerning a liquid flowing through a closed channel. A considerable degree of success was obtained using a boundary immobilization scheme with finite differences, especially with regard to predicting the solid/liquid interface location as a function of the physical parameters involved and determining the critical conditions for the situations of steady-state and blockage.

APPENDIX 1

VAX 11/780 FORTRAN Codes for the Classical Stefan Problem

This appendix contains a listing of the five FORTRAN codes used in the first chapter relating to the classical one-dimensional Stefan problem (note that the FORTRAN implemented here is for use on DEC VAX 11/780 machines and several features are non-standard FORTRAN). The variables used follow those in the text for the most part but for clarity a list is given at the beginning of each code. Variables repeated in later listings will be defined once.

Appendix 1.1 A "Direct" Implicit Difference Scheme

```
VARIABLES : x(0:100)        Spacial grid
            u(0:100,2)      Discrete temperature distribution
            iu(0:100,2)     Temperature iterates at each time step
            s(2)            Interface location
            tmax            Maximum time of simulation
            t0              Initial time for numerical scheme
            dt              Time step
            dx              Mesh size
            r               Stability parameter
            alp,bet,gam     Constants in finite difference equations
            the0,the1                       Do.
            lam             Freezing parameter
            s15aef          Error function (NAG library)
            cpu             CPU time
```

```
      program implicit_gauss_1d

*********************************************************************
* PROGRAM : This program solves the classic Stefan (melting) problem *
*           using the standard implicit finite-difference scheme for *
*           the  temperature distribution and solving the  resulting *
*           equations by an iterative procedure (Gauss-Seidel).  The *
*           interface location is  calculated 'explicitly'. This and *
*           the temperature calculations are in an overall iterative *
*           loop.  A comparison  with the exact solution  at tmax is *
*           produced.                                                *
*********************************************************************

      character term

      integer   efi

      real*8    t0,dt,tmax,x(0:100),s(2),u(0:100,2),eps,the0,the1,
     c          lam,s15aef,r,alp,bet,gam,cpu,dx,iu(0:100,2),ss
```

```
+-----------------------------------------------------------------------+
| INPUT DATA ON CHANNEL 5 & CALCULATE CONSTANTS                         |
+-----------------------------------------------------------------------+

      write(6,101)
      read(5,200)term
      iout = 6
      if( term.eq.'f'.or.term.eq.'F' ) iout = 1
      write(6,102)
      read(5,*)t0,dt,tmax,dx,efi

      do 1 i = 0,100
    1 x(i) = dflotj( i )*dx

      ifail = 0
      lam   = 1.24012527d0
      eps   = 10.0**(-efi)
      r     = dt/(dx*dx)
      it    = jifix( sngl(tmax/dt-t0/dt+dt/2.0) )

+-----------------------------------------------------------------------+
| SET UP THE 'SMOOTH' INITIAL CONDITIONS                                |
+-----------------------------------------------------------------------+

      s(1) = lam*dsqrt( t0 )
      im   = s(1)/dx
      the0 = ( s(1)-im*dx )/dx
      do 2 i = 1,im
    2 u(i,1) = 1.0 - s15aef( x(i)/(2.0*dsqrt(t0)),ifail )
     c               /s15aef( lam/2.0,ifail )

+-----------------------------------------------------------------------+
| BOUNDARY CONDITIONS                                                   |
+-----------------------------------------------------------------------+

      do 10 m = 1,2
   10 u(0,m) = 1.0

+-----------------------------------------------------------------------+
| IMPLICIT SCHEME (with iteration)                                      |
+-----------------------------------------------------------------------+
| INCREMENT TIME                                                        |
+-----------------------------------------------------------------------+

   21 call actim( icpul )

      s(2) = s(1)

      do 23 m = 1,it

      iover = 0

+-----------------------------------------------------------------------+
| INITIAL APPROXIMATION FOR THE ITERATES                                |
+-----------------------------------------------------------------------+

      do 24 i = 1,im
```

```
   24  iu(i,1) = u(i,1)

+-----------------------------------------------------------------------+
| ITERATION LOOP                                                        |
+-----------------------------------------------------------------------+

   25  iover = iover+1
       itota = itota+1

+-----------------------------------------------------------------------+
| NEW INTERFACE LOCATION                                                |
+-----------------------------------------------------------------------+

       ss = s(1) - (dt/dx)*( -((1.0+the0)*iu(im,1))/the0
     c                       +(the0*iu(im-1,1))/(1.0+the0) )
       im   = ss/dx
       the1 = ( ss-im*dx )/dx
       alp  = 6.0/( the1*( 1.0 + the1 )*( 2.0 + the1 ) )
       bet  = ( 2.0*(2.0 - the1) )/(1.0 + the1)
       gam  = (the1 - 1.0)/(the1 + 2.0)

+-----------------------------------------------------------------------+
| NEW TEMPERATURE ITERATES                                              |
+-----------------------------------------------------------------------+

       iu(1,2) = ( r*iu(2,1) + u(1,1) + r )/(1.0 + 2.0*r)
       do 27 i = 2,im-1
   27  iu(i ,2) = ( r*iu(i-1,2) + r*iu(i+1,1) + u(i,1) )/(1.0+2.0*r)
       iu(im,2) = ( r*gam*iu(im-2,2) + r*bet*iu(im-1,2) + u(im,1) )
     c           /(1.0 + r*(alp + bet + gam))

+-----------------------------------------------------------------------+
| CHECK CONVERGENCE                                                     |
+-----------------------------------------------------------------------+

       do 34 i = 1,im
       if( dabs( iu(i,2)-iu(i,1) ).gt.eps ) go to 35
   34  continue

       if( dabs( ss-s(2) ).le.eps ) go to 30

+-----------------------------------------------------------------------+
| RESET FOR NEW ITERATION                                               |
+-----------------------------------------------------------------------+

   35  do 37 i = 1,im
   37  u(i,1) = u(i,2)

       s(2) = ss
       the0 = the1

       go to 25

+-----------------------------------------------------------------------+
| RESET FOR NEXT TIME STEP                                              |
+-----------------------------------------------------------------------+

   30  s(1) = ss
```

```
      s(2) = ss
      im   = s(1)/dx
      do 32 i = 1,im
   32 u(i,1) = iu(i,2)
      if( iover.gt.mio ) mio = iover
   23 the0 = the1
```

```
+-----------------------------------------------------------------------+
| CALCULATE EXACT SOLUTION AT TMAX                                      |
+-----------------------------------------------------------------------+
```

```
   22 call actim( icpu2 )

      s(2) = lam*dsqrt( tmax )
      ix   = s(2)/dx

      do 17 i = 0,ix
   17 u(i,2) = 1.0 - s15aef( x(i)/(2.0*dsqrt(tmax)),ifail )
     c              /s15aef( lam/2.0,ifail )

      if( im.gt.ix ) then
           do 20 i = ix+1,im
   20      u(i,2) = 0.0
      end if
```

```
+-----------------------------------------------------------------------+
| OUTPUT NUMERICAL & EXACT SOLUTIONS AT TMAX                            |
+-----------------------------------------------------------------------+
```

```
      cpu = dflotj( icpu2 - icpu1 )/100.0d0

      icol = 13
      if( term.eq.'V'.or.term.eq.'v' ) icol = 7

      if( term.eq.'p'.or.term.eq.'P' ) then
           write(6,103)
           read(5,200)term
      end if

      write(iout,104)
      write(iout,105)r
      write(iout,106)it,dx,eps,dt,t0,tmax

      im  = jmax0( im,ix )
      ilo = 0
      ihi = jmin0( icol,im )

   13 write(iout,107)(x(i)  ,i = ilo,ihi)
      write(iout,108)(u(i,1),i = ilo,ihi)
      write(iout,109)(u(i,2),i = ilo,ihi)

      if( ihi.eq.im ) go to 14
      ilo = ihi + 1
      ihi = ihi + jmin0( icol,im-ihi-1 ) + 1
      go to 13

   14 write(iout,110)s(1),s(2)
      write(iout,111)cpu
```

```
      write(iout,112)mio
      write(iout,113)itota
```

```
+-----------------------------------------------------------------------+
| FORMAT STATEMENTS                                                     |
+-----------------------------------------------------------------------+
```

```
  101  format(1x,'Output to (V)ideo,(P)rinter,(F)ile ? ',$)
  102  format(1x,'Input t0,dt,tmax,dx,efi                ',$)
  103  format(1x,'Position Paper - Press RET',$)
  104  format(40x,'"DIRECT" IMPLICIT TECHNIQUE'/1x)
  105  format(1x,'GAUSS-SEIDEL ITERATION (r=',f4.2,')'/1x)
  106  format(1x,'No. time-steps (it) :',i6/
     c        1x,'Mesh size      (dx) :',f7.4/
     c        1x,'Accuracy      (eps) :',f11.8/
     c        1x,'Time-Step      (dt) :',f12.9/
     c        1x,'Initial Time   (t0) :',f7.4/1x/
     c        1x,'TIME =',f7.4)
  107  format(1x/1x/1x,' x -> ',14(f8.4,1x))
  108  format(1x/1x,'Num T ',14f9.6)
  109  format(1x/1x,'Ext T ',14f9.6)
  110  format(1x/1x/1x,'Interface Location (Numerical)   :',f13.8/
     c        1x,19x,'(Exact)        :',f13.8/1x)
  111  format(1x,'CPU time ( numerical procedure ) :',f10.3,
     c          ' secs'/1x)
  112  format(1x,'Max. iterations required ( mio ) :',i6)
  113  format(1x,'Total iterations       ( itota ) :',i6)
  200  format(A1)

       stop
       end
```

Appendix 1.2 A Variable Grid Scheme

```
VARIABLES : ds            Exact mesh size at tmax
            dsdt          Interface velocity
            n             Fixed number of mesh segments
```

```
      program var_grid_1d

*********************************************************************
* PROGRAM : This program solves the classic Stefan (melting) problem *
*           using a variable grid size with a constant number, N, of *
*           intervals. An explicit finite  difference  approximation *
*           is used and a comparison with the exact solution at tmax *
*           is given.                                                *
*********************************************************************

      character term

      real*8    u(0:200,2),x(0:200),s15aef,ds,dx,dt,t0,tmax,r,cpu,
     c          dsdt,s,lam

+-------------------------------------------------------------------+
| INPUT DATA & CALCULATE CONSTANTS                                  |
+-------------------------------------------------------------------+

      write(6,101)
      read(5,200)term
      iout = 6
      if( term.eq.'f'.or.term.eq.'F' ) iout = 1
      write(6,102)
      read(5,*)t0,tmax,n

      lam  = 1.24012527d0
      x(0) = 0.0d0
      x(n) = lam*dsqrt(t0)
      dx   = x(n)/dflotj(n)

      write(6,103)0.5d0*dx*dx
      write(6,104)
      read(5,*)dt

      do 1 i = 1,n-1
      x(i)   = dflotj( i )*dx
    1 u(i,1) = 1.0d0 - s15aef( x(i)/(2.0d0*dsqrt(t0)),ifail )
     c                /s15aef( lam/2.0d0,ifail )

      it = jifix( sngl((tmax-t0)/dt + dt/2.0) )

+-------------------------------------------------------------------+
| BOUNDARY CONDITIONS                                               |
+-------------------------------------------------------------------+

      u(0,1) = 1.0d0
      u(n,1) = 0.0d0
```

```
+-----------------------------------------------------------------------+
| TIME INCREMENT                                                        |
+-----------------------------------------------------------------------+

      call actim( icpul )

      do 2 m = 1,it

      r = dt/(dx*dx)

+-----------------------------------------------------------------------+
| INTERFACE VELOCITY                                                    |
+-----------------------------------------------------------------------+

      dsdt = -( u(n-2,1) - 4.0d0*u(n-1,1) )/( 2.0d0*dx )

+-----------------------------------------------------------------------+
| NEW TEMPERATURES                                                      |
+-----------------------------------------------------------------------+

      do 3 i = 1,n-1
    3 u(i,2) = u(i,1)
     c       + ( dt*x(i)*dsdt*( u(i+1,1) - u(i-1,1) ) )
     c        /( x(n)*2.0d0*dx )
     c       + r*( u(i+1,1) - 2.0d0*u(i,1) + u(i-1,1) )

+-----------------------------------------------------------------------+
| NEW MESH SIZE & SPACIAL NODES                                         |
+-----------------------------------------------------------------------+

      x(n) = x(n) + dt*dsdt
      dx   = x(n)/dflotj(n)
      do 4 i = 1,n-1
    4 x(i) = dflotj(i)*dx

+-----------------------------------------------------------------------+
| RESET FOR NEXT TIME STEP                                              |
+-----------------------------------------------------------------------+

      do 5 i = 1,n-1
    5 u(i,1) = u(i,2)

    2 continue

+-----------------------------------------------------------------------+
| EXACT SOLUTION AT TMAX                                                |
+-----------------------------------------------------------------------+

      call actim( icpu2 )

      cpu = dflotj( icpu2 - icpul )*0.01d0

      s  = lam*dsqrt( tmax )
      ds = s/dflotj(n)

      do 7 i = 0,n
    7 u(i,2) = 1.0d0 - s15aef( i*ds/(2.0d0*dsqrt(tmax)),ifail )
```

```
      c                 /s15aef( lam/2.0d0,ifail )
```

OUTPUT

```
      icol = 13
      if( term.eq.'v'.or.term.eq.'V' ) icol = 7

      if( term.eq.'p'.or.term.eq.'P' ) then
           write(6,105)
           read(5,200)term
      end if

      write(iout,106)
      write(iout,107)it,n,dt,t0,tmax

      ilo = 0
      ihi = jmin0( icol,n )

   10 write(iout,108)(i*ds  ,i = ilo,ihi)
      write(iout,109)(x(i)  ,i = ilo,ihi)
      write(iout,110)(u(i,1),i = ilo,ihi)
      write(iout,111)(u(i,2),i = ilo,ihi)

      if( ihi.eq.n ) go to 11
      ilo = ihi + 1
      ihi = ihi + jmin0( icol,n-ihi-1 ) + 1
      go to 10

   11 write(iout,112)cpu
```

FORMAT STATEMENTS

```
  101 format(1x,'Ouput to (V)ideo,(P)rinter,(F)ile ? ',$)
  102 format(1x,'Input t0,tmax,n                    ',$)
  103 format(1x/1x,'For r <= 0.5, t <= ',f16.8 )
  104 format(1x/1x,'Input dt                          ',$)
  105 format(1x,'Position Paper - Press RET')
  106 format(40x,'VARIABLE SPACE GRID'/1x)
  107 format(1x,'No. time steps (it) :',i6/
     c       1x,'No. space nodes (n) :',i6/
     c       1x,'Time step      (dt) :',f11.8/
     c       1x,'Initial time   (t0) :',f7.4/1x/
     c       1x,'TIME =',f7.4)
  108 format(1x/1x/1x,'Ext x ',14f9.6)
  109 format(1x/1x,'Num x ',14f9.6)
  110 format(1x/1x,'Num T ',14f9.6)
  111 format(1x/1x,'Ext T ',14f9.6)
  112 format(1x/1x/1x,'CPU Time :',f8.3,' secs')
  200 format(A)

      stop
      end
```

Appendix 1.3 A Boundary Immobilization Scheme

```
VARIABLES : mu(0:100)     Immobilized co-ordinate
            dmu           Immobilized mesh size
            z(2)          Interface location
            ut0(0:10)     Initial temperature (for comparison)
            dzdt          Interface velocity


      program immob_explicit

*********************************************************************
* PROGRAM : This program solves the classic Stefan (melting) problem *
*           using a boundary immobilisation technique. An explicit *
*           finite-difference scheme is used to approximate the *
*           resulting equations. A comparison with the exact *
*           solution at tmax is produced.                          *
*********************************************************************

      character term

      real*8    u(0:100,2),mu(0:100),t0,tmax,dmu,dt,lam,dx,z(2),
     c          x(0:100),ut0(0:10),cpu,r,time,s15aef,dzdt,dz

+-------------------------------------------------------------------+
| INPUT DATA & CALCULATE CONSTANTS                                  |
+-------------------------------------------------------------------+

      write(6,101)
      read(5,200)term
      iout = 6
      if( term.eq.'f'.or.term.eq.'F' ) iout = 1
      write(6,102)
      read(5,*)t0,dt,tmax,dmu

      ifail = 0
      lam   = 1.24012527d0
      in    = 1.0/dmu
      it    = jifix( sngl(tmax/dt-t0/dt+dt/2.0d0) )
      r     = dt/(dmu*dmu)

      do 1 i = 0,in
    1 mu(i) = dflotj( i )*dmu

+-------------------------------------------------------------------+
| CALCULATE INITIAL TEMP. DISTRIBUTION AT INCS. OF DMU = 0.1        |
+-------------------------------------------------------------------+

      write(iout,103)
      z(1)   = lam*lam*t0
      x(0)   = 0.0
      ut0(0) = 1.0
      do 19 i = 1,10
      x(i)   = dflotj(i)*0.1d0*dsqrt( z(1) )
   19 ut0(i) = 1.0 - s15aef( x(i)/(2.0d0*dsqrt(t0)),ifail )
     c              /s15aef( lam/2.0d0,ifail )
```

```
      write(iout,104)t0

+-----------------------------------------------------------------------+
| BOUNDARY CONDITIONS                                                   |
+-----------------------------------------------------------------------+

      do 3 m = 1,2
      u(0 ,m) = 1.0
    3 u(in,m) = 0.0

+-----------------------------------------------------------------------+
| SET INITIAL TEMPERATURE & INTERFACE                                   |
+-----------------------------------------------------------------------+

      do 2 i = 1,in-1
    2 u(i,1) = 1.0 - s15aef( mu(i)*lam*0.5d0,ifail )
     c               /s15aef( lam*0.5d0,ifail )

+-----------------------------------------------------------------------+
| EXPLICIT SCHEME                                                       |
+-----------------------------------------------------------------------+
| TIME INCREMENT                                                        |
+-----------------------------------------------------------------------+

      call actim( icpul )

      do 27 m = 1,it

+-----------------------------------------------------------------------+
| INTERFACE VELOCITY                                                    |
+-----------------------------------------------------------------------+

      dzdt = -( -4.0d0*u(in-1,1) + u(in-2,1) )/dmu

+-----------------------------------------------------------------------+
| NEW TEMPERATURE                                                       |
+-----------------------------------------------------------------------+

      do 31 i = 1,in-1
   31 u(i,2) = u(i,1)
     c       + r*( (mu(i)*dzdt*dmu*(u(i+1,1)-u(i-1,1)))/(4.0d0*z(1))
     c            +(u(i+1,1)-2.0d0*u(i,1)+u(i-1,1))/z(1) )

      z(1) = z(1) + dt*dzdt

+-----------------------------------------------------------------------+
| RESET FOR NEXT TIME STEP                                              |
+-----------------------------------------------------------------------+

   38 do 40 i = 1,in-1
   40 u(i,1) = u(i,2)

   27 continue
```

```
+-----------------------------------------------------------------------+
| CONVERT TO 'FREE' CO-ORDINATES                                        |
+-----------------------------------------------------------------------+

      call actim( icpu2 )

      cpu = dflotj( icpu2 - icpu1 )*0.01d0

      z(1) = dsqrt( z(1) )

      do 14 i = 0,in
   14 x(i) = mu(i)*z(1)

      dx = dmu*z(1)

+-----------------------------------------------------------------------+
| EXACT SOLUTION AT TMAX                                                |
+-----------------------------------------------------------------------+

      z(2) = lam*dsqrt( tmax )
      dz   = z(2)/dflotj(in)

      do 15 i = 1,in
   15 u(i,2) = 1.0 - s15aef( (dflotj(i)*dz)/(2.0*dsqrt(tmax)),ifail )
     c              /s15aef( lam/2.0,ifail )

+-----------------------------------------------------------------------+
| OUTPUT                                                                |
+-----------------------------------------------------------------------+

      icol = 13
      if( term.eq.'V'.or.term.eq.'v' ) icol = 7

      if( term.eq.'p'.or.term.eq.'P' ) then
          write(6,105)
          read(5,200)term
      end if

      write(iout,106)it,dmu,dx,dt,tmax

      ilo = 0
      ihi = jmin0( icol,in )

   23 write(iout,107)(x(i)  ,i = ilo,ihi)
      write(iout,108)(u(i,1),i = ilo,ihi)
      write(iout,109)(u(i,2),i = ilo,ihi)

      if( ihi.eq.in ) go to 24
      ilo = ihi + 1
      ihi = ihi + jmin0( icol,in-ihi-1 ) + 1
      go to 23

   24 write(iout,110)z(1),z(2)
      write(iout,111)cpu
```

FORMAT STATEMENTS

```
  101  format(1x,'Output to (V)ideo,(P)rinter,(F)ile ? ',$)
  102  format(1x,'Input t0,dt,tmax,dmu                 ',$)
  103  format(40x,'BOUNDARY IMMOBILIZATION'/1X)
  104  format(1x,'Initial Time (t0) :',f7.4/1x)
  105  format(1x,'Position Paper - Press RET',$)
  106  format(1x,'No. Time steps     (it) :',i6/
     c        1x,'Immob. Mesh size  (dmu) :',f7.4/
     c        1x,'"Free" Mesh size   (dx) :',f11.8/
     c        1x,'Time-step          (dt) :',f11.8/1x/
     c        1x,'TIME =',f7.4)
  107  format(1x/1x/1x,' x -> ',14f9.6)
  108  format(1x/1x,'Num T ',14f9.6)
  109  format(1x/1x,'Ext T ',14f9.6)
  110  format(1x/1x,'Interface Location (Numerical) :',f13.8/
     c        1x,19x,'(Exact)     :',f13.8)
  111  format(1x/1x,'CPU Time (numerical procedure) :',f8.3,' secs')
  200  format(A1)

       stop
       end
```

Appendix 1.4 The Isotherm Migration Method

```
VARIABLES : temp(0:100)   Temperature co-ordinates
            dtemp         Temperature mesh size
            ds            Contains maximum allowed time step
            dt1,dt2       Lower & upper time steps over tmax
            acc           Accuracy of  Newton's iteration scheme
                          for obtaining exact isotherm locations


      program iso_exp_1d

*************************************************************************
* PROGRAM : This program solves the classic Stefan (melting) problem *
*           using the 'Isotherm Migration' technique. The non-linear *
*           equations  are explicitly differenced.  For the explicit *
*           scheme an estimate of the time-step is made (for reasons *
*           of stability), thus  the  time  step changes  with time. *
*           A comparison  with the exact solution  at tmax is given. *
*************************************************************************

      character term

      integer*4 afi,efi

      real*8    x(0:100,2),temp(0:100),t0,dt,tmax,dtemp,lam,time,
     c          ds,cpu,acc,dt1,dt2

+-----------------------------------------------------------------------+
| INPUT DATA & CALCULATE CONSTANTS                                      |
+-----------------------------------------------------------------------+

      write(6,101)
      read(5,200)term
      iout = 6
      if( term.eq.'f'.or.term.eq.'F' ) iout = 1
      write(6,102)
      read(5,*)t0,tmax,dtemp,afi

      lam = 1.24012527
      acc = 10.0**(-afi)
      in  = 1.0/dtemp

      do 1 i = 0,in
    1 temp(i) = dflotj( i )*dtemp

+-----------------------------------------------------------------------+
| SET INITIAL CONDITIONS (LOCATIONS OF INITIAL ISOTHERMS)               |
+-----------------------------------------------------------------------+

      x(0,1) = lam*dsqrt( t0 )
      do 2 i = 1,in-1
      call xroot( temp,i,1,x,t0,lam,acc )
    2 continue
```

```
+-----------------------------------------------------------------------+
| BOUNDARY CONDITIONS                                                   |
+-----------------------------------------------------------------------+

      do 14 m = 1,2
   14 x(in,m) = 0.0

+-----------------------------------------------------------------------+
| EXPLICIT ALGORITHM                                                    |
+-----------------------------------------------------------------------+
| TIME INCREMENT                                                        |
+-----------------------------------------------------------------------+

      time = t0
      it   = 0

      call actim( icpul )

    3 it = it + 1

+-----------------------------------------------------------------------+
| ESTIMATE TIME STEP AND NEW TIME                                       |
+-----------------------------------------------------------------------+

      dt = 0.125*( x(2,1) - x(0,1) )*( x(2,1) - x(0,1) )
      do 24 i = 3,in
      ds = 0.125*( x(i,1) - x(i-2,1) )*( x(i,1) - x(i-2,1) )
   24 dt = dminl( dt,ds )

      if( it.eq.1 ) dt1 = dt
      if( time + dt.gt.tmax ) then
           dt   = tmax - time
           time = tmax
           else
           time = time + dt
           dt2 = dt
      end if

+-----------------------------------------------------------------------+
| NEW ISOTHERM LOCATIONS                                                |
+-----------------------------------------------------------------------+

    9 x(0,2) = x(0,1) - ( dt*dtemp )
     c                 /( -0.5*x(2,1) + 2.0*x(1,1) - 1.5*x(0,1) )
      do 4 i = 1,in-1
    4 x(i,2) = x(i,1) + ( 4.0*dt*(x(i+1,1)-2.0*x(i,1)+x(i-1,1)) )
     c                 /( (x(i+1,1)-x(i-1,1))*(x(i+1,1)-x(i-1,1)) )

+-----------------------------------------------------------------------+
| RESET FOR THE NEXT TIME STEP                                          |
+-----------------------------------------------------------------------+

    7 do 10 i = 0,in-1
   10 x(i,1) = x(i,2)
      if( time.ge.tmax ) go to 25
      go to 3
```

```
+-----------------------------------------------------------------------+
| EXACT SOLUTION AT TIME TMAX                                           |
+-----------------------------------------------------------------------+

   25  call actim( icpu2 )

       x(0,2) = lam*dsqrt( tmax )
       do 11 i = 1,in-1
       call xroot( temp,i,2,x,tmax,lam,acc )
   11  continue

+-----------------------------------------------------------------------+
| OUTPUT                                                                |
+-----------------------------------------------------------------------+

       cpu = dflotj( icpu2 - icpu1 )/100.0d0

       icol = 13
       if( term.eq.'V'.or.term.eq.'v' ) icol = 7

       if( term.eq.'p'.or.term.eq.'P' ) then
            write(6,103)
            read(5,200)term
       end if

       write(iout,104)
       write(iout,105)it
       write(iout,106)dt1,dt2
       write(iout,107)dtemp,t0,acc,tmax

       ilo = 0
       ihi = jmin0( icol,in )

   13  write(iout,108)(temp(i),i = ilo,ihi)
       write(iout,109)(x(i,1) ,i = ilo,ihi)
       write(iout,110)(x(i,2) ,i = ilo,ihi)

       if( ihi.eq.in ) go to 12
       ilo = ihi + 1
       ihi = ihi + jmin0( icol,in-ihi-1 ) + 1
       go to 13
   12  continue

       write(iout,111)cpu

+-----------------------------------------------------------------------+
| FORMAT STATEMENTS                                                     |
+-----------------------------------------------------------------------+

  101  format(1x,'Output to (V)ideo,(P)rinter,(F)ile ? ',$)
  102  format(1x,'Input t0,tmax,dtemp,afi              ',$)
  103  format(1x,'Position Paper - Press RET',$)
  104  format(40x,'ISOTHERM MIGRATION'/1x)
  105  format(1x,'No. time-steps   (it) :',i6)
  106  format(1x,'Lower time-step (dt1) :',f11.8/
     c       1x,'Upper time-step (dt2) :',f11.8)
  107  format(1x,'Mesh size     (dtemp) :',f7.4/
```

```
     c        1x,'Initial time     (t0) :',f7.4/
     c        1x,'<Root> Accuracy (acc) :',f11.8/1x/
     c       1x,'TIME =',f7.4)
  108  format(1x/1x/1x,'Temp. ',14(f8.4,1x))
  109  format(1x/1x,'Num x ',14f9.6)
  110  format(1x/1x,'Ext x ',14f9.6)
  111  format(1x/1x/1x,'CPU time ( numerical procedure ) :',f10.3,
     c                ' secs')
  200  format(A1)

       stop
       end

       subroutine xroot( te,i,j,x,ti,lam,acc )

*************************************************************************
* SUBROUTINE : This  routine obtains  the position (x) of particular *
*              isotherms using the exact solution. Newtons iteration *
*              scheme is used.                                        *
*************************************************************************

       real*8  te(0:100),x(0:100,2),ti,lam,s15aef,x01aaf,pi,f,df,
     c         y(2),acc

       ifail = 0
       pi    = 4.0*datan( 1.0d0 )
       y(1)  = x(i,1)

    2  f  = 1.0 - te(i) - s15aef( y(1)/(2.0*dsqrt(ti)),ifail )
     c                  /s15aef( lam/2.0,ifail )

       df = - dexp( -(y(1)*y(1))/(4.0*ti) )
     c       /( s15aef( lam/2.0,ifail )*dsqrt( pi*ti ) )

       y(2) = y(1) - f/df

       if( dabs( y(2)-y(1) ).le.acc ) go to 1

       y(1) = y(2)
       go to 2

    1  x(i,j) = y(2)

       return
       end
```

Appendix 1.5 The Enthalpy Method with the Fully Explicit or Hopscotch Difference Scheme

VARIABLES : e(0:200,2) Discrete enthalpy distribution
sche Numerical scheme (fully explicit or hopscotch)
ient (C)entre or (R)ight initial enthalpy definition

```
      program enth_hops_exp_1d

************************************************************************
* PROGRAM : This program solves the classic Stefan (melting) problem *
*           using the Enthalpy Method. The resulting equation is *
*           approximated and solved using the hopscotch or explicit *
*           finite-difference scheme. A comparison with the exact *
*           solution at tmax is produced. An estimate of the CPU *
*           time used by the numerical algorithm is also produced. *
************************************************************************

      character term,sche,ient

      real*8    u(0:200,2),e(0:200,2),x(0:200),s15aef,lam,dx,dt,t0,r,
     c          tmax,s(2),cpu

+----------------------------------------------------------------------+
| INPUT DATA ON CHANNEL 5                                              |
+----------------------------------------------------------------------+

      write(6,100)
      read(5,200)term
      write(6,101)
      read(5,200)sche
      write(6,102)
      read(5,200)ient
      iout = 6
      if(term.eq.'f'.or.term.eq.'F') iout = 1
      write(6,103)
      read(5,*)t0,dt,tmax,dx

+----------------------------------------------------------------------+
| CALCULATE CONSTANTS                                                  |
+----------------------------------------------------------------------+

      do 1 i = 0,200
    1 x(i) = dflotj( i )*dx

      ifail = 0
      r     = dt/(dx*dx)
      lam   = 1.24012527d0
      it    = jifix( sngl(tmax/dt-t0/dt+dt/2.0d0) )
```

```
+-----------------------------------------------------------------------+
| INITIAL INTERFACE, TEMPERATURE & ENTHALPY DISTRIBUTIONS               |
+-----------------------------------------------------------------------+

      s(1) = lam*dsqrt( t0 )
      im   = s(1)/dx
      do 2 i = 1,im
      u(i,1) = 1.0d0 - s15aef( x(i)/(2.0d0*dsqrt(t0)),ifail )
     c                /s15aef( lam/2.0d0,ifail )
    2 e(i,1) = u(i,1) + 1.0d0

+-----------------------------------------------------------------------+
| INITIAL ENTHALPY ADJACENT TO THE INTERFACE                            |
+-----------------------------------------------------------------------+

      if( ient.eq.'c'.or.ient.eq.'C' ) then
          if( s(1).lt.( dflotj(im)+0.5d0 )*dx ) then
              e(im  ,1) = ( s(1) - ( dflotj(im)-0.5d0 )*dx )/dx
              e(im+1,1) = 0.0d0
              else
              e(im  ,1) = u(im,1) + 1.0d0
              e(im+1,1) = ( s(1) - ( dflotj(im)+0.5d0 )*dx )/dx
          end if
          else
          e(im+1,1) = ( s(1)-im*dx )/dx
      end if

+-----------------------------------------------------------------------+
| BOUNDARY CONDITIONS                                                   |
+-----------------------------------------------------------------------+

      do 3 m = 1,2
    3 u(0,m) = 1.0d0

      if( sche.eq.'H'.or.sche.eq.'h' ) go to 14

+-----------------------------------------------------------------------+
| EXPLICIT NUMERICAL SCHEME                                             |
+-----------------------------------------------------------------------+

      call actim( icpul )

      do 4 m = 1,it

      do 5 i = 1,200
      e(i,2) = e(i,1) + r*( u(i-1,1) - 2.0d0*u(i,1) + u(i+1,1) )
      if( e(i,2).gt.1.0d0 ) then
          u(i,2) = e(i,2) - 1.0d0
          else
          u(i,2) = 0.0d0
      end if
      if( e(i,2).eq.0.0d0 ) go to 6
    5 continue
```

```
+-----------------------------------------------------------------------+
| RESET FOR NEXT TIME STEP                                              |
+-----------------------------------------------------------------------+

    6  do 7 i = 1,200
       u(i,1) = u(i,2)
    7  e(i,1) = e(i,2)

    4  continue

       call actim( icpu2 )

       go to 15

+-----------------------------------------------------------------------+
| HOPSCOTCH NUMERICAL SCHEME                                            |
+-----------------------------------------------------------------------+

   14  call actim( icpu1 )

       do 8 m = 1,it

       iodd = jmod( m,2 )

+-----------------------------------------------------------------------+
| Explicit Algorithm                                                    |
+-----------------------------------------------------------------------+

       do 9 i = iodd+1,200,2
       e(i,2) = e(i,1) + r*( u(i-1,1) - 2.0d0*u(i,1) + u(i+1,1) )
       if( e(i,2).gt.1.0d0 ) then
            u(i,2) = e(i,2) - 1.0d0
            else
            u(i,2) = 0.0d0
       end if
       if( e(i,2).eq.0.0d0 ) go to 10
    9  continue

+-----------------------------------------------------------------------+
| Implicit Algorithm                                                    |
+-----------------------------------------------------------------------+

   10  do 11 i = 2-iodd,200,2
       e(i,2) = e(i,1) + r*( u(i-1,2) + u(i+1,2) )
       if( e(i,2).gt.1.0d0 ) then
            e(i,2) = ( e(i,2) + 2.0d0*r )/( 1.0d0 + 2.0d0*r )
            u(i,2) = e(i,2) - 1.0d0
            else
            u(i,2) = 0.0d0
       end if
       if( e(i,2).eq.0.0d0 ) go to 12
   11  continue
```

```
+-----------------------------------------------------------------------+
| RESET FOR NEXT TIME STEP                                              |
+-----------------------------------------------------------------------+

   12  do 13 i = 1,200
       u(i,1) = u(i,2)
   13  e(i,1) = e(i,2)

    8  continue

       call actim( icpu2 )

+-----------------------------------------------------------------------+
| ESTIMATE OF 'NUMERICAL' INTERFACE LOCATION AT TMAX                    |
+-----------------------------------------------------------------------+

   15  do 16 i = 200,1,-1
       if( e(i,1).gt.0.0d0 ) then
            if( ient.eq.'c'.or.ient.eq.'C' ) then
                 s(1) = ( dflotj(i) - 0.5d0 + e(i,1) )*dx
                 else
                 s(1) = ( dflotj(i) - 1.0d0 + e(i,1) )*dx
            end if
            go to 17
       end if
   16  continue
   17  im = s(1)/dx

+-----------------------------------------------------------------------+
| EXACT SOLUTION AT TMAX                                                |
+-----------------------------------------------------------------------+

       s(2) = lam*dsqrt( tmax )
       ix   = s(2)/dx

       do 18 i = 1,ix
   18  u(i,2) = 1.0d0 - s15aef( x(i)/(2.0d0*dsqrt(tmax)),ifail )
     c                /s15aef( lam/2.0d0,ifail )

       if( im.gt.ix ) then
            do 21 i = ix+1,im
   21       u(i,2) = 0.0d0
       end if

+-----------------------------------------------------------------------+
| OUTPUT                                                                |
+-----------------------------------------------------------------------+

       cpu  = dflotj( icpu2 - icpu1 )/100.0d0

       ico1 = 13
       if( term.eq.'V'.or.term.eq.'v' ) ico1 = 7

       if( term.eq.'p'.or.term.eq.'P' ) then
            write(6,104)
            read(5,200)term
       end if
```

```
      write(iout,105)ient
      if( sche.eq.'H'.or.sche.eq.'h' ) write(iout,106)r
      if( sche.eq.'E'.or.sche.eq.'e' ) write(iout,107)r
      write(iout,108)it,dx,dt,t0,tmax

      im  = jmax0( ix,im )
      ilo = 0
      ihi = jmin0( icol,im )

   19 write(iout,109)(x(i)  ,i = ilo,ihi)
      write(iout,110)(u(i,1),i = ilo,ihi)
      write(iout,111)(u(i,2),i = ilo,ihi)

      if( ihi.eq.im ) go to 20
      ilo = ihi + 1
      ihi = ihi + jmin0( icol,im-ihi-1 ) + 1
      go to 19

   20 write(iout,112)s(1),s(2)
      write(iout,113)cpu
```

+--+
| FORMAT STATEMENTS |
+--+

```
  100 format(1x,'O/P to (V)ideo,(P)rinter,(F)ile  ? ',$)
  101 format(1x,'(E)xplicit or (H)opscotch scheme ? ',$)
  102 format(1x,'(C)entre or (R)ight enthalpies   ? ',$)
  103 format(1x,'I/P t0,dt,tmax,dx                 ',$)
  104 format(1x,'Position Paper - Press RET')
  105 format(40x,'THE ENTHALPY METHOD (',A1,')'/1x)
  106 format(1x,'HOPSCOTCH ALGORITHM',2x,'(r=',f4.2,')'/1x)
  107 format(1x,'EXPLICIT ALGORITHM',2x,'(r=',f4.2,')'/1x)
  108 format(1x,'No. Time-steps (it) :',i6/
     c       1x,'Mesh size      (dx) :',f7.4/
     c       1x,'Time-step      (dt) :',f12.9/
     c       1x,'Initial Time   (t0) :',f7.4/1x/
     c       1x,'TIME =',f7.4)
  109 format(1x/1x/1x,' x -> ',14(f8.4,1x))
  110 format(1x/1x,'Num T ',14f9.6)
  111 format(1x/1x,'Ext T ',14f9.6)
  112 format(1x/1x/1x,'Interface Location (Numerical) :',f13.8/
     c       1x,19x,'(Exact)     :',f13.8/1x)
  113 format(1x,'CPU Time (Numerical Procedure) :',f8.3,' secs')
  200 format(A1)

      stop
      end
```

Appendix 2.1

Local Temperature Behaviour about the Singular Point for the 'Element' Problem

With regard to figure 2.3 we consider the local (normalised) temperature $T(r,\theta,t)$ to be given as

$$T(r,\theta,t) = e^{-\alpha^2 t} R(r)\,\psi(\theta) + w(r,\theta) \qquad - (A2.1.1)$$

subject to

$$T(r,\theta=0,t) = 1.0$$

$$T(r=0,\theta,t) = 0.0$$

$$\frac{\partial T(r,\theta=\pi,t)}{\partial\theta} = 0.0$$

$w(r,\theta)$ is the steady-state solution and the first term on the right-hand side of (A2.1.1) represents the transient nature of the solution.

Considering the steady-state solution first we note that $w(r,\theta)$ satisfies

$$\frac{\partial^2 w}{\partial r^2} + \frac{1}{r}\frac{\partial w}{\partial r} + \frac{1}{r^2}\frac{\partial^2 w}{\partial\theta^2} = 0$$

$$w(r,0,t) = 1.0$$

$$\frac{\partial w(r,\pi,t)}{\partial\theta} = 0.0$$

Assuming the solution $w(r,\theta) = 1 + \delta(r)\,\beta(\theta)$ we easily obtain

$$\beta = a_1\cos(\lambda_1\theta) + a_2\sin(\lambda_1\theta)$$

$$\delta = r^{\lambda_1} \quad \text{(for a finite solution as } r \to 0\text{)}$$

Using the boundary conditions we obtain

$$w = \sum_{k=0}^{\infty} C_k \, r^{(2k+1)\lambda} \sin[(2k+1)\lambda\theta] \qquad - (A2.1.2)$$

where $\lambda = \pi/2\theta_o$ (θ_o is angle of the 'wedge'). Substituting (A2.1.1) into the transient conduction equation

$$\frac{\partial^2 T}{\partial r^2} + \frac{1}{r}\frac{\partial T}{\partial r} + \frac{1}{r^2}\frac{\partial^2 T}{\partial \theta^2} = \frac{\partial T}{\partial t} \qquad - (A2.1.3)$$

yields

$$\psi \frac{d^2 R}{dr^2} + \frac{\psi}{r}\frac{dR}{dr} + \frac{R}{r^2}\frac{d^2\psi}{d\theta^2} = -\alpha^2 R \psi \qquad - (A2.1.4)$$

from which we have the solutions

$$\psi = X_1 \cos(\omega\theta) + X_2 \sin(\omega\theta)$$

$$R = y \, J_\omega(\kappa r)$$

where ω is the separation constant of (A2.1.4). Using these solutions and (A2.1.2) we may construct the solution $T(r,\theta,t)$. Applying the boundary conditions provides the solution

$$T(r,\theta,t) = 1 + \sum_{k=0}^{\infty} A_k \, e^{-\kappa^2 t} \, J_{(2k+1)\lambda}(\alpha r) \sin\{(2k+1)\lambda\theta\}$$

$$+ \sum_{k=0}^{\infty} C_k \, r^{(2k+1)\lambda} \sin\{(2k+1)\lambda\theta\}$$

If we then sum over all j roots of the Bessel function, note that $\theta_o = \pi$ ($\Rightarrow \lambda = 1/2$) and use the expansion

$$J_n(z) = \sum_{m=0}^{\infty} \frac{(-1)^m (z/2)^{2m+n}}{m!\Gamma(n+m+1)}$$

we obtain the general solution

$$T(r,\theta,t) = 1 + \sum_{j=0}^{\infty}\sum_{k=0}^{\infty} A_{kj}\, e^{-\alpha_j^2 t} \sum_{m=0}^{\infty} \frac{(-1)^m(\alpha_j r/2)^{2m+k+1/2}}{m!\Gamma(n+m+1/2)} \sin\{(k+1/2)\theta\}$$
$$+ \sum_{k=0}^{\infty} C_k\, r^{k+1/2} \sin\{(k+1/2)\theta\}$$

where Γ is the gamma function.

To obtain the first few terms in this series solution we group the co-efficients of the terms $r^{\frac{1}{2}}$, $r^{\frac{3}{2}}$, $r^{\frac{5}{2}}$ etcetera which yields

$$T(r,\theta,t) = 1 + a_0 r^{\frac{1}{2}} \sin(\theta/2) + a_1 r^{\frac{3}{2}} \sin(3\theta/2)$$
$$+ [a_2 \sin(5\theta/2) - b_0 \sin(\theta/2)] r^{\frac{5}{2}}$$
$$+ [a_3 \sin(7\theta/2) - b_1 \sin(3\theta/2)] r^{\frac{7}{2}} + O(r^{\frac{9}{2}})$$

[equation (A2.1.5)] where the a_i are functions of time. For example

$$a_0(t) = \sum_{j=0}^{\infty} A_{0j}\, e^{-\alpha_j^2 t} \frac{(\alpha_j/2)^{\frac{1}{2}}}{\Gamma(3/2)} + C_0$$

To obtain a relation between the co-efficients in (A2.1.5) we substitute the series back into the differential equation (A2.1.3) and equate like terms in r and θ. This provides the solution (equation (16)) given in chapter II.

APPENDIX 2.2

VAX 11/780 FORTRAN Code for the Explicit Difference Solution of the 'Test' Problem of Section 2.3

As with the one-dimensional codes this FORTRAN implementation is local in certain respects and several non-standard features are included. Some of the main variables used in the program are described briefly.

VARIABLES :	t(0:100,0:100,2)	Discrete, 2-D temperature distribution
	e(0:100,0:100)	Discrete, 2-D enthalpy distribution
	ub(0:100)	U locations of interface on mesh
	vb(0:100)	V locations of interface on mesh
	du,dv,dt	Mesh sizes & time step
	it,ft	Initial & final time for simulation

```
      program test_problem

*************************************************************************
* PROGRAM : This  program computes  a  finite difference solution to  *
*           the  test  problem  posed  in  section  2.3. The  fully  *
*           algorithm  is  used  with  or  without  a  correction to  *
*           account for the  singular point. The  numerical solution  *
*           solution  may be compared to the analytic solution  that  *
*           exists for the problem.                                   *
*************************************************************************

      character*4 sing

      real*8      it,ft,dt,du,dv,lam,c1,s15aef,umax,vmax,y,loc(5),z,
     c            ub(0:100),vb(0:100),ubmax,u(0:100),v(0:100),phi,x,r,
     c            t(0:100,0:100,2),e(0:100,0:100)

+-----------------------------------------------------------------------+
| INPUT DATA                                                            |
+-----------------------------------------------------------------------+

      write(6,101)
      read(5,*)it,ft,dt
      write(6,102)
      read(5,*)du
      write(6,103)
      read(5,*)iser
      sing = 'NO'
      if( iser.eq.1 ) sing = 'YES'

+-----------------------------------------------------------------------+
| CALCULATE CONSTANTS                                                   |
+-----------------------------------------------------------------------+

      call lambda( lam )
```

```
      ift   = jifix( sngl((ft-it)/dt) )
      ie    = jifix( sngl(1.0d0/du) )
      ifail = 0
      dv    = du
      cl    = s15aef( lam,ifail )
      y     = 2.0*lam*dsqrt(ft)
      umax  = dcosh(y)
      vmax  = dsinh(y)
      imax  = jifix( sngl(umax/du) ) + 5
      jmax  = jifix( sngl(vmax/dv) ) + 5
      r     = dt/(du*du)

      call setlocals( du )

      do 1 i = 0,imax
    1 u(i) = dflotj(i)*du

      do 2 j = 0,jmax
    2 v(j) = dflotj(j)*dv
```

```
+-----------------------------------------------------------------------+
| BOUNDARY CONDITION                                                    |
+-----------------------------------------------------------------------+
```

```
      do 3 i = 0,ie
      t(i,0,1) = 1.0
    3 t(i,0,2) = 1.0
```

```
+-----------------------------------------------------------------------+
| CALCULATE INITIAL INTERFACE LOCATION                                  |
+-----------------------------------------------------------------------+
```

```
      y  = 2.0*lam*dsqrt(it)
      ib = jifix( sngl(dcosh(y)/du) )

      call interface( du,dv,ub,vb,nb,y )
```

```
+-----------------------------------------------------------------------+
| CALCULATE INITIAL TEMPERATURE & 'INTERNAL' ENTHALPIES                 |
+-----------------------------------------------------------------------+
```

```
      do 4 i = 0,ib
      z  = dtanh(y)*dsqrt( dcosh(y)*dcosh(y) - u(i)*u(i) )
      jb = jifix( sngl(z/dv) )

      do 4 j = jmax0(0,1-i/ie),jb
      x = u(i)*u(i) + v(j)*v(j)
     c         + dsqrt( (u(i)+1.0)*(u(i)+1.0) + v(j)*v(j) )
     c          *dsqrt( (u(i)-1.0)*(u(i)-1.0) + v(j)*v(j) )
      phi       = 0.5*dlog( x + dsqrt( x*x - 1.0 ) )
      t(i,j,1) = 1.0 - s15aef( phi/(2.0*dsqrt(it)),ifail )/cl
    4 e(i,j)   = 1.0 + t(i,j,1)
```

```
+-----------------------------------------------------------------------+
| CALCULATE 'INTERFACE' ENTHALPIES                                      |
+-----------------------------------------------------------------------+
```

```
      call interenth( du,dv,ub,vb,nb,y,e )
```

```
+-----------------------------------------------------------------------+
| TIME STEP                                                             |
+-----------------------------------------------------------------------+

      call actim( icpul )

      do 5 m = 1,ift

+-----------------------------------------------------------------------+
| EXPLICIT CALCULATIONS                                                 |
+-----------------------------------------------------------------------+

      call explicit_test( ie,imax,jmax,r,u,v,t,e )

      if( iser.eq.1 ) call sercoeffs( du,ie,t,e )

+-----------------------------------------------------------------------+
| RESET TEMPERATURES                                                    |
+-----------------------------------------------------------------------+

      do 7 j = 0,jmax
      do 7 i = 0,imax
    7 t(i,j,1) = t(i,j,2)

    5 continue

      call actim( icpu2 )

      cpu =  0.01*(icpu2 - icpul)

+-----------------------------------------------------------------------+
| OUTPUT                                                                |
+-----------------------------------------------------------------------+
| Analytic Isotherms (FOR004) for plotting                              |
+-----------------------------------------------------------------------+

      y = 2.0*lam*dsqrt(ft)

      call isotherm_location( ft,lam,loc,y )

      do 8 i = 1,5
      um = dcosh( loc(i) )
      im = jifix( sngl(um/du) )
      write(4,104)im+2,dflotj(i-1)*0.2
      do 9 k = 0,im
      vm = dtanh(loc(i))*dsqrt( um*um - dflotj(k)*du*dflotj(k)*du )
    9 write(4,105)k*du,vm
    8 write(4,105)um,0.0

+-----------------------------------------------------------------------+
| Numerical Spot Temperatures (FOR002) for contour plots                |
+-----------------------------------------------------------------------+

      do 21 j = jmax,0,-1
      kj = j
      if( t(0,kj,1).gt.0.0d0 ) go to 22
   21 continue
```

```
22  do 23 i = imax,0,-1
    ki = i
    if( t(ki,0,1).gt.0.0d0 ) go to 24
23  continue
24  write(2,0106)ki+1,kj+1,du,dv
    do 12 j = kj,0,-1
    do 13 i = 0,ki
13  if( t(i,j,1).lt.0.0d0 ) t(i,j,1) = 0.0
12  write(2,107)(t(i,j,1),i = 0,ki)
```

Observable Numerical Data (FOR003) for inspection

```
    icol = 0
    jcol = jmin0( 21,ki )
16  do 14 j = kj,0,-1
    do 15 i = icol,jcol
    if( t(i,j,1).gt.0.0d0 ) then
        write(3,108)t(i,j,1)
        else
        write(3,109)
        go to 14
    end if
15  continue
    write(3,109)
14  continue
    if( jcol.lt.ki ) then
        icol = jcol + 1
        jcol = jmin0( jcol+22,ki )
        pause 'set paper'
        go to 16
    end if
```

Analytic Spot Temperatures (FOR001) for contour plots

```
    um = dcosh(y)
    vm = dsinh(y)
    ki = jifix( sngl(um/du) )
    kj = jifix( sngl(vm/dv) )
    write(1,106)ki+1,kj+1,du,dv
    il = 0
    do 10 j = kj,0,-1
    if( j.eq.0 ) il = ie + 1
    do 11 i = il,ki
    x = u(i)*u(i) + v(j)*v(j)
  c   + dsqrt( (u(i)+1.0)*(u(i)+1.0) + v(j)*v(j) )
  c    *dsqrt( (u(i)-1.0)*(u(i)-1.0) + v(j)*v(j) )
    phi      = 0.5*dlog( x + dsqrt( x*x - 1.0 ) )
    t(i,j,2) = 1.0 - s15aef( phi/(2.0*dsqrt(ft)),ifail)/c1
    if( t(i,j,2).lt.0.0d0 ) t(i,j,2) = 0.0
11  continue
10  write(1,107)(t(i,j,2),i = 0,ki)
```

Observable Analytic Data (FOR007) for inspection

```
      icol = 0
      jcol = jmin0( 21,ki )
   36 do 34 j = kj,0,-1
      do 35 i = icol,jcol
      if( t(i,j,2).gt.0.0d0 ) then
           write(7,108)t(i,j,2)
           else
           write(7,109)
           go to 34
      end if
   35 continue
      write(7,109)
   34 continue
      if( jcol.lt.ki ) then
           icol = jcol + 1
           jcol = jmin0( jcol+22,ki )
           pause 'set paper'
           go to 36
      end if

      write(6,110)du,dt,r,it,ft,sing
      write(6,111)cpu
```

FORMAT STATEMENTS

```
  101 format(1x,'Input (Initial,Final,Inc.) Times > ',$)
  102 format(1x,'Input Mesh Size                   > ',$)
  103 format(1x,'Sing. ? (1/0)                     > ',$)
  104 format(1x,i3,f4.1)
  105 format(1x,2f11.8)
  106 format(1x,2i3,2f5.2)
  107 format(1x,12f11.8)
  108 format('+',f6.3,$)
  109 format(1x)
  110 format(1x/
     c       1x,'Mesh size = ',f5.2/
     c       1x,'Time step = ',f11.8/
     c       1x,'Stab. par = ',f7.4/
     c       1x,'Ini. time = ',f5.2/
     c       1x,'Fin. time = ',f5.2/
     c       1x,'Singularity taken into account ? ',A4)
  111 format(1x/1x,'CPU Time = ',f7.2,' seconds'/1x)

      stop
      end
```

```
      subroutine lambda( lam )

*************************************************************************
* SUBROUTINE : This routine calculates the freezing paramater lambda    *
*              for the non-dimensional (1-D) Stefan problem with the    *
*              initial temperature  equal to the  fusion temperature    *
*              (see page 43) using the bisection method.                *
*************************************************************************

      integer*4 ifail

      real*8    lam,l1,l2,f,s15aef

      ifail = 0
      l1    = 0.0
      l2    = 5.0
    1 lam   = (l1 + l2)/2.0
      f     = dexp( -lam*lam )
     c      - 2.0*dsqrt(datan(1.0d0))*lam*s15aef( lam,ifail )

      if( dabs(f).le.1.0d-12 ) return

      if( f.lt.0.0d0 ) then
          l2 = lam
          else
          l1 = lam
      end if

      go to 1

      end

      subroutine interface( du,dv,ub,vb,nb,y )

*************************************************************************
* SUBROUTINE : This  routine calculates  the initial location of the    *
*              interface  and in particular  the  points at which it    *
*              crosses the finite difference grid.                      *
*************************************************************************

      integer*4 ibmax,k,j1,j2,nb

      real*8    du,dv,ub(0:100),vb(0:100),ubmax,vbmax,u,v,y

      ubmax = dcosh(y)
      vbmax = dsinh(y)
      ibmax = jifix( sngl(ubmax/du) )

      ub(0) = 0.0
      vb(0) = vbmax

      k  = 0
      j2 = jifix( sngl(vbmax/dv) )

      do 1 i = 1,ibmax+1

      j1 = j2
```

```
      u  = dflotj(i)*du

      if( u.gt.ubmax ) u = ubmax

      v  = dtanh(y)*dsqrt( ubmax*ubmax - u*u )
      j2 = jifix( sngl(v/du) )

      if( j1.eq.j2 ) then
           k = k + 1
           ub(k) = u
           vb(k) = v
           else
           do 2 j = j1,j2+1,-1
           k = k + 1
           vb(k) = dflotj(j)*dv
    2      ub(k) = dsqrt( vbmax*vbmax - vb(k)*vb(k) )/dtanh(y)
           k = k + 1
           ub(k) = u
           vb(k) = v
      end if

    1 continue

      nb = k

      return
      end

      subroutine setlocals( du )

*************************************************************************
* SUBROUTINE : This  routine defines  the locations  of the 3 points  *
*              used to  match the  local temperature  series. The  4  *
*              points  to  be corrected  are also  located  and  the  *
*              co-efficients of the a(t) are set up.                  *
*************************************************************************

      real*8 du,a(3),d(3),pi,sum(3,3),r(4),th(4),mat(4,3)

      common sum,mat

      pi = 4.0*datan(1.0d0)

      d(1) = 2.0*du
      d(2) = dsqrt(8.0d0)*du
      d(3) = d(2)

      a(1) = pi
      a(2) = 0.75*pi
      a(3) = 0.25*pi

      do 1 k = 1,3
      sum(k,1) = dsqrt(d(k))*dsin(0.5*a(k))
      sum(k,2) = d(k)*sum(k,1)
    1 sum(k,3) = d(k)*dsqrt(d(k))*dsin(1.5*a(k))

      r(1)  = du
      r(2)  = dsqrt(2.0d0)*du
```

```
      r(3)  = du
      r(4)  = dsqrt(2.0d0)*du

      th(1) = pi
      th(2) = 0.75*pi
      th(3) = 0.50*pi
      th(4) = 0.25*pi

      do 2 k = 1,4
      mat(k,1) = dsqrt(r(k))*dsin(0.5*th(k))
      mat(k,2) = r(k)*mat(k,1)
    2 mat(k,3) = r(k)*dsqrt(r(k))*dsin(1.5*th(k))

      return
      end

      subroutine interenth( du,dv,ub,vb,nb,y,e )

*************************************************************************
* SUBROUTINE : This  routine calculates  the initial enthalpy at the *
*              points adjacent to and outside the liquid region (see *
*              notes on page 49).                                     *
*************************************************************************

      integer*4 nb,i,j

      real*8    du,dv,ub(0:100),vb(0:100),y,e(0:100,0:100),u1,u2,
     c          v1,v2,bint,v,u,umax

      umax = dcosh(y)

      u1 = ub(0)
      v1 = vb(0)

      j = jifix( sngl(v1/dv) ) + 1

      e(0,j) = ( dflotj(j)*dv - v1 )/dv

      do 1 n = 1,nb
      u1 = ub(n-1)
      v1 = vb(n-1)
      u2 = ub(n)
      v2 = vb(n)
      i  = jifix( sngl(u2/du) )
      u  = dflotj(i)*du

      if( u.ne.u2 ) i = i + 1

      j  = jifix( sngl(v1/dv) )
      v  = dflotj(j)*dv

      if( v.ne.v1 ) j = j + 1

      v  = dflotj(j-1)*dv

      call bintegral( u1,u2,bint,y )

    1 e(i,j)  = (bint + (u1-dflotj(i-1)*du)*dv - (u2-u1)*v)/(du*dv)
```

```
      e(0,nb) = (dflotj(nb)*du - umax)/du

      return
      end

      subroutine sercoeffs( du,ie,t,e )

*****************************************************************
* SUBROUTINE : This routine sets up a set  of simultaneous equations *
*              and solves  them  for the co-efficients  of the local *
*              temperature  series. The  corrected temperatures  and *
*              corresponding enthalpies are then evaluated.         *
*****************************************************************

      real*8 coef(3),rhs(3),t(0:100,0:100,2),du,pi,sum(3,3),
     c       mat(4,3),pcof(4),e(0:100,0:100)

      common sum,mat

      pi = 4.0*datan(1.0d0)

      rhs(1) = t(ie+2,0,2) - 1.0
      rhs(2) = t(ie+2,2,2) - 1.0
      rhs(3) = t(ie-2,2,2) - 1.0

      call gauss( 2,3,rhs,coef,sum )

      do 2 n = 1,4
      pcof(n) = 1.0
      do 2 k = 1,3
    2 pcof(n) = pcof(n) + mat(n,k)*coef(k)

      t(ie+1,0,2) = pcof(1)
      t(ie+1,1,2) = pcof(2)
      t(ie  ,1,2) = pcof(3)
      t(ie-1,1,2) = pcof(4)

      e(ie+1,0) = t(ie+1,0,2) + 1.0
      e(ie+1,1) = t(ie+1,1,2) + 1.0
      e(ie  ,1) = t(ie  ,1,2) + 1.0
      e(ie-1,1) = t(ie-1,1,2) + 1.0

      return
      end

      subroutine bintegral( ulo,uhi,bint,y )

*****************************************************************
* SUBROUTINE : This routine evaluates the integral of  the interface *
*              over an element du = uhi - ulo.                      *
*****************************************************************

      real*8 ulo,uhi,bint,y,z1,z2

      z1 = dasin( ulo/dcosh(y) )
      z2 = dasin( uhi/dcosh(y) )
```

```
      bint = 0.25*dsinh(2.0*y)
     c       *(z2 + 0.5*dsin(2.0*z2) - z1 - 0.5*dsin(2.0*z1))

      return
      end

      subroutine gauss(degree,degpl,rhs,coeff,mat)

*************************************************************************
* SUBROUTINE : This  routine solves  a system of algebraic equations *
*              using Gauss elimination with partial pivoting.         *
*************************************************************************

      integer degree,degpl

      real*8  rhs(3),coeff(3),sum(3,3),dump,factor,total,mat(3,3)

      do 111 i = 1,3
      do 111 j = 1,3
  111 sum(i,j) = mat(i,j)

      do 10 k = 1,degree
      kplus1 = k+1
      l = k
      do 11 i = kplus1,degpl

      if( dabs(sum(i,k)).le.dabs(sum(l,k)) ) go to 11

      l = i
   11 continue

      if( l.le.k ) go to 12

      do 13 j = k,degpl
      dump    = sum(k,j)
      sum(k,j) = sum(l,j)
   13 sum(l,j) = dump

      dump    = rhs(k)
      rhs(k)  = rhs(l)
      rhs(l)  = dump

   12 do 10 i = kplus1,degpl
      factor  = sum(i,k)/sum(k,k)
      sum(i,k) = 0.0d0

      do 14 j = kplus1,degpl
   14 sum(i,j) = sum(i,j)-factor*sum(k,j)

   10 rhs(i) = rhs(i) - factor*rhs(k)

      coeff(degpl) = rhs(degpl)/sum(degpl,degpl)
      i = degree
   16 iplus1 = i+1
      total = 0.0d0

      do 15 j = iplus1,degpl
   15 total = total+sum(i,j)*coeff(j)
```

```
      coeff(i) = (rhs(i)-total)/sum(i,i)
      i = i-1

      if( i.gt.0 ) go to 16

      return
      end

      subroutine explicit_test( ie,imax,jmax,r,u,v,t,e )

*************************************************************************
* SUBROUTINE : This routine executes one pass of the fully  explicit *
*              finite difference algorithm over the solution region. *
*************************************************************************

      integer*4 ie,imax,jmax

      real*8    r,u(0:100),v(0:100),t(0:100,0:100,2),e(0:100,0:100),
     c          x

      do 1 j = 0,jmax

      if( j.eq.0 ) then

           do 2 i = ie+1,imax
           x = dabs( u(i)+1.0d0 )*dabs( u(i)-1.0d0 )
           e(i,0) = e(i,0)
     c           + r*x*( t(i+1,0,1) + t(i-1,0,1)
     c                 + 2.0*t(i,1,1) - 4.0*t(i,0,1) )
           if( e(i,0).gt.1.0 ) then
                t(i,0,2) = e(i,0) - 1.0
                else
                t(i,0,2) = 0.0
                go to 1
           end if
    2      continue

           else

           x = 1.0 + v(j)*v(j)
           e(0,j) = e(0,j)
     c           + r*x*( 2.0*t(1,j,1) + t(0,j+2,2) + t(0,j-1,1)
     c                 - 4.0*t(0,j,1) )
           if( e(0,j).gt.1.0d0 ) then
                t(0,j,2) = e(0,j) - 1.0
                else
                t(0,j,2) = 0.0
                if( e(0,j).le.0.0d0 ) return
           end if

           do 3 i = 1,imax
           x = dsqrt((u(i)+1.0)*(u(i)+1.0) + v(j)*v(j))
     c        *dsqrt((u(i)-1.0)*(u(i)-1.0) + v(j)*v(j))
           e(i,j) = e(i,j)
     c           + r*x*( t(i+1,j,1) + t(i-1,j,1)
     c                 + t(i,j,1+1) + t(i,j,1-1)
     c                 - 4.0*t(i,j,1) )
```

```
            if( e(i,j).gt.1.0 ) then
                 t(i,j,2) = e(i,j) - 1.0
                 else
                 t(i,j,2) = 0.0
                 if( e(i,j).le.0.0d0 ) go to 1
            end if
    3       continue
        end if


    1   continue

        return
        end

        subroutine isotherm_location( time,lam,loc,y )

*************************************************************************
* SUBROUTINE : This routine calculates the analytic  locations of  4 *
*              isotherms in the  one-dimensional problem to be  used *
*              for  calculating  the corresponding  ellipses  in the *
*              'test' problem.                                       *
*************************************************************************


        integer*4 ifail

        real*8    time,t,lam,loc(5),l1,l2,g,y,s15aef,c1

        loc(1) = y
        ifail  = 0
        c1     = s15aef( lam,ifail )

        do 1 i = 2,5
        t  = dflotj(i-1)*0.2
        l1 = 0.0
        l2 = y
    2   loc(i) = (l1+l2)/2.0
        g = s15aef( loc(i)/(2.0*dsqrt(time)),ifail ) - c1*( 1.0 - t )

        if( dabs(g).le.1.0d-12 ) go to 1

        if( g.lt.0.0d0 ) then
             l1 = loc(i)
             else
             l2 = loc(i)
        end if

        go to 2

    1   continue
     loc(5) = y

        return
        end
```

APPENDIX 2.3

VAX 11/780 FORTRAN Code for the Hopscotch Difference Solution of the 'Test' Problem of Section 2.3

This appendix lists the explicit and implicit hopscotch algorithms for the 'Test' problem. A 'fast' coding procedure is used, due to the nature of the hopscotch scheme, in which the newly calculated temperatures immeadiately over-write the old values as with the enthalpy. This reduces the dimension of the temperature array and also reduces indexing time. In addition there is no requirement for a piece of 'resetting' code. The program of appendix 2.2 is used deleteting loop 7, replacing the section 'EXPLICIT CALCULATIONS' with the first piece of code listed here and replacing the subroutine EXPLICIT_TEST with the two routines listed below. Also, the dimension of 't' is reduced.

(Calling procedure)

```
+-----------------------------------------------------------------------+
| HOPSCOTCH CALCULATIONS                                                |
+-----------------------------------------------------------------------+

c Explicit Algorithm

      call hops_exp_test( m,ie,imax,jmax,r,u,v,t,e )

c Implicit Algorithm

      call hops_imp_test( m,ie,imax,jmax,r,u,v,t,e )
```

(Hopscotch subroutines)

```
      subroutine hops_exp_test( m,ie,imax,jmax,r,u,v,t,e )

**********************************************************************
* SUBROUTINE : This routine executes one pass of the explicit *
*              hopscotch algorithm over the solution region for the *
*              'Test' Problem.                                       *
**********************************************************************

      integer*4 m,ie,imax,jmax,is

      real*8    r,u(0:100),v(0:100),t(0:100,0:100),e(0:100,0:100),x

      do 1 j = 0,jmax

      is = jmod( j+m,2 )

      if( j.eq.0 ) then
           do 2 i = ie+2-is,imax,2
```

```
            x = dabs( u(i)+1.0d0 )*dabs( u(i)-1.0d0 )
            e(i,0) = e(i,0)
     c             + r*x*( t(i+1,0) + t(i-1,0)
     c                   + 2.0*t(i,1) - 4.0*t(i,0) )
            if( e(i,0).gt.1.0 ) then
                 t(i,0) = e(i,0) - 1.0
                 else
                 t(i,0) = 0.0
                 go to 1
            end if
        2   continue

            else

            do 3 i = is,imax,2
            x = dsqrt( (u(i)+1.0)*(u(i)+1.0) + v(j)*v(j) )
     c         *dsqrt( (u(i)-1.0)*(u(i)-1.0) + v(j)*v(j) )
            if( i.eq.0 ) then
                 e(0,j) = e(0,j)
     c                 + r*x*( 2.0*t(1,j) + t(0,j+1)
     c                       + t(0,j-1) - 4.0*t(0,j) )
                 else
                 e(i,j) = e(i,j)
     c                 + r*x*( t(i+1,j) + t(i-1,j)
     c                       + t(i,j+1) + t(i,j-1)
     c                       - 4.0*t(i,j) )
            end if
            if( e(i,j).gt.1.0 ) then
                 t(i,j) = e(i,j) - 1.0
                 else
                 t(i,j) = 0.0
                 if( e(i,j).le.0.0d0 ) then
                      if( i.eq.0 ) return
                      go to 1
                 end if
            end if
        3   continue
        end if

        1 continue

        return
        end

        subroutine hops_imp_test( m,ie,imax,jmax,r,u,v,t,e )

*************************************************************************
* SUBROUTINE : This routine executes one pass of the implicit *
*              hopscotch algorithm over the solution region for the *
*              'Test' Problem.                                      *
*************************************************************************

        integer*4 m,ie,imax,jmax,is

        real*8    r,u(0:100),v(0:100),t(0:100,0:100),e(0:100,0:100),x

        do 1 j = 0,jmax
```

```
      is = jmod( j+m+1,2 )

      if( j.eq.0 ) then
           do 2 i = ie+2-is,imax,2
           x = dabs( u(i)+1.0d0 )*dabs( u(i)-1.0d0 )
           e(i,0) = e(i,0)
     c           + r*x*( t(i+1,0) + t(i-1,0) + 2.0*t(i,1) )
           if( e(i,0).gt.1.0 ) then
                e(i,0) = ( e(i,0) + 4.0*r*x )/( 1.0 + 4.0*r*x )
                t(i,0) = e(i,0) - 1.0
                else
                t(i,0) = 0.0
                go to 1
           end if
    2      continue

           else

           do 3 i = is,imax,2
           x = dsqrt( (u(i)+1.0)*(u(i)+1.0) + v(j)*v(j) )
     c        *dsqrt( (u(i)-1.0)*(u(i)-1.0) + v(j)*v(j) )
           if( i.eq.0 ) then
                e(0,j) = e(0,j)
     c                + r*x*(2.0*t(1,j) + t(0,j+1) + t(0,j-1))
                else
                e(i,j) = e(i,j)
     c                + r*x*( t(i+1,j) + t(i-1,j)
     c                      + t(i,j+1) + t(i,j-1) )
           end if
           if( e(i,j).gt.1.0 ) then
                e(i,j) = ( e(i,j) + 4.0*r*x )/( 1.0 + 4.0*r*x )
                t(i,j) = e(i,j) - 1.0
                else
                t(i,j) = 0.0
                if( e(i,j).le.0.0d0 ) then
                     if( i.eq.0 ) return
                     go to 1
                end if
           end if
    3      continue
      end if

    1 continue

      return
      end
```

APPENDIX 2.4

VAX 11/780 FORTRAN Code for the Hopscotch Difference Solution of the 'Element' Problem of Section 2.2

This appendix lists the explicit and implicit hopscotch algorithms for the 'Element' problem. As in appendix 2.3 a 'fast' coding procedure is adopted and the program of appendix 2.2 is used with the following amendments. Loop 7 is deleted and the section 'EXPLICIT CALCULATIONS' is replaced with the calling procedure of appendix 2.3. The output on channels 001, 004 and 007 is deleted since there is no analytic solution to this problem. For a similar reason the subroutine ISOTHERM_LOCATION is deleted. The subroutine EXPLICIT_TEST is replaced with the two hopscotch routines listed below. Since we are now using 4 points for matching the local series expansion the modified versions of the routines SETLOCALS and SERCOEFFS are listed.

```
      subroutine setlocals( du )

**********************************************************************
* SUBROUTINE : This  routine defines  the locations  of the 4 points *
*              used to  match the  local temperature  series. The  4 *
*              points  to  be corrected  are also  located  and  the *
*              co-efficients of the a(t) are set up.                 *
**********************************************************************

      real*8 du,a(4),d(4),pi,sum(4,4),r(4),th(4),mat(4,4)

      common sum,mat

      pi = 4.0*datan(1.0d0)

      d(1) = 2.0*du
      d(2) = dsqrt(8.0d0)*du
      d(3) = d(2)
      d(4) = d(1)

      a(1) = pi
      a(2) = 0.75*pi
      a(3) = 0.25*pi
      a(4) = 0.50*pi

      do 1 k = 1,4
      sum(k,1) = dsqrt(d(k))*dsin(0.5*a(k))
      sum(k,2) = d(k)*dsqrt(d(k))*dsin(1.5*a(k))
      sum(k,3) = d(k)*d(k)*sum(k,1)
    1 sum(k,4) = d(k)*d(k)*dsqrt(d(k))*dsin(2.5*a(k))

      r(1)  = du
      r(2)  = dsqrt(2.0d0)*du
      r(3)  = du
      r(4)  = dsqrt(2.0d0)*du
```

```
      th(1) = pi
      th(2) = 0.75*pi
      th(3) = 0.50*pi
      th(4) = 0.25*pi

      do 2 k = 1,4
      mat(k,1) = dsqrt(r(k))*dsin(0.5*th(k))
      mat(k,2) = r(k)*dsqrt(r(k))*dsin(1.5*th(k))
      mat(k,3) = r(k)*r(k)*mat(k,1)
    2 mat(k,4) = r(k)*mat(k,2)

      return
      end

      subroutine sercoeffs( du,ie,t,e )

*************************************************************************
* SUBROUTINE : This routine sets up a set  of simultaneous equations *
*              and solves  them  for the co-efficients  of the local *
*              temperature  series. The  corrected temperatures  and *
*              corresponding enthalpies are then evaluated.          *
*************************************************************************

      integer*4 ie

      real*8    coef(4),rhs(4),t(0:100,0:100),du,pi,sum(4,4),
     c          mat(4,4),pcof(4),e(0:100,0:100)

      common    sum,mat

      pi = 4.0*datan(1.0d0)

      rhs(1) = t(ie+2,0) - 1.0
      rhs(2) = t(ie+2,2) - 1.0
      rhs(3) = t(ie-2,2) - 1.0
      rhs(4) = t(ie  ,2) - 1.0

      call gauss( 3,4,rhs,coef,sum )

      do 2 n = 1,4
      pcof(n) = 1.0
      do 2 k = 1,4
    2 pcof(n) = pcof(n) + mat(n,k)*coef(k)

      t(ie+1,0) = pcof(1)
      t(ie+1,1) = pcof(2)
      t(ie  ,1) = pcof(3)
      t(ie-1,1) = pcof(4)

      e(ie+1,0) = t(ie+1,0) + 1.0
      e(ie+1,1) = t(ie+1,1) + 1.0
      e(ie  ,1) = t(ie  ,1) + 1.0
      e(ie-1,1) = t(ie-1,1) + 1.0

      return
      end
```

```
      subroutine hops_exp_test( m,ie,imax,jmax,r,u,v,t,e )

*************************************************************************
* SUBROUTINE : This  routine  executes  one  pass  of  the  explicit  *
*              hopscotch algorithm  over the solution region for the  *
*              'Element' Problem (sections 2.2 & 2.5).                *
*************************************************************************

      integer*4 m,ie,imax,jmax,is

      real*8    r,u(0:100),v(0:100),t(0:100,0:100),e(0:100,0:100)

      do 1 j = 0,jmax

      is = jmod( j+m,2 )

      if( j.eq.0 ) then
           do 2 i = ie+2-is,imax,2
           e(i,0) = e(i,0)
     c           + r*( t(i+1,0) + t(i-1,0)
     c               + 2.0*t(i,1) - 4.0*t(i,0) )
           if( e(i,0).gt.1.0 ) then
                t(i,0) = e(i,0) - 1.0
                else
                t(i,0) = 0.0
                go to 1
           end if
    2      continue

           else

           do 3 i = is,imax,2
           if( i.eq.0 ) then
                e(0,j) = e(0,j)
     c                + r*( 2.0*t(1,j) + t(0,j+1)
     c                    + t(0,j-1) - 4.0*t(0,j) )
                else
                e(i,j) = e(i,j)
     c                + r*( t(i+1,j) + t(i-1,j)
     c                    + t(i,j+1) + t(i,j-1)
     c                    - 4.0*t(i,j) )
           end if
           if( e(i,j).gt.1.0 ) then
                t(i,j) = e(i,j) - 1.0
                else
                t(i,j) = 0.0
                if( e(i,j).le.0.0d0 ) then
                     if( i.eq.0 ) return
                     go to 1
                end if
           end if
    3      continue
      end if

    1   continue

      return
```

```
      end

      subroutine hops_imp_test( m,ie,imax,jmax,r,u,v,t,e )

*************************************************************************
* SUBROUTINE : This  routine  executes  one  pass  of  the  implicit    *
*              hopscotch algorithm  over the solution region for the    *
*              'Element' Problem (sections 2.2 & 2.5).                  *
*************************************************************************

      integer*4 m,ie,imax,jmax,is

      real*8    r,u(0:100),v(0:100),t(0:100,0:100),e(0:100,0:100)

      do 1 j = 0,jmax

      is = jmod( j+m+1,2 )

      if( j.eq.0 ) then
           do 2 i = ie+2-is,imax,2
           e(i,0) = e(i,0) + r*( t(i+1,0) + t(i-1,0) + 2.0*t(i,1) )
           if( e(i,0).gt.1.0 ) then
                e(i,0) = ( e(i,0) + 4.0*r )/( 1.0 + 4.0*r )
                t(i,0) = e(i,0) - 1.0
                else
                t(i,0) = 0.0
                go to 1
           end if
    2      continue

           else

           do 3 i = is,imax,2
           if( i.eq.0 ) then
                e(0,j) = e(0,j)
     c                 + r*( 2.0*t(1,j) + t(0,j+1) + t(0,j-1) )
                else
                e(i,j) = e(i,j)
     c                 + r*( t(i+1,j) + t(i-1,j)
     c                     + t(i,j+1) + t(i,j-1) )
           end if
           if( e(i,j).gt.1.0 ) then
                e(i,j) = ( e(i,j) + 4.0*r )/( 1.0 + 4.0*r )
                t(i,j) = e(i,j) - 1.0
                else
                t(i,j) = 0.0
                if( e(i,j).le.0.0d0 ) then
                     if( i.eq.0 ) return
                     go to 1
                end if
           end if
    3      continue
      end if

    1 continue

      return
      end
```

APPENDIX 3.1

Evaluation of the Contour Integrals for the Elliptic Pure Conduction Problem

(Note that equation and chapter references, unless otherwise stated, refer to McLachlan (1947)).

(i) Line Segments BC and FA

We require to show that $\lim_{R\to\infty} \int_{BC} = 0$ and $\lim_{R\to\infty} \int_{FA} = 0$.

In the limit $R \to \infty$ (for fixed 'a', see figure 3.2) we let $s = Re^{i\theta}$, $\theta_o \leqslant \theta \leqslant \pi$, on the arc BC. We define the function $\phi_m(Re^{i\theta})$ by

$$\phi_m(Re^{i\theta}) = (-1)^m A_0^{(2m)} \frac{Fek_{2m}(\xi, -k^2)}{Fek_{2m}(\xi_o, -k^2)} \frac{ce_{2m}(\eta, -k^2)}{Re^{i\theta}}$$

and hence

$$\left|\lim_{R\to\infty} \int_{BC}\right| = \left| \sum_{m=0}^{\infty} \lim_{R\to\infty} \int_{\theta_o}^{\pi} e^{Rte^{i\theta}} \phi_m(Re^{i\theta})\, iRe^{i\theta}\, d\theta \right|$$

$$\leqslant \sum_{m=0}^{\infty} \lim_{R\to\infty} \int_{\theta_o}^{\pi} \left| e^{Rte^{i\theta}} \phi_m(Re^{i\theta})\, iRe^{i\theta} \right| d\theta$$

$$= \sum_{m=0}^{\infty} \lim_{R\to\infty} \int_{\theta_o}^{\pi} Re^{Rt\cos\theta} \left|\phi_m(Re^{i\theta})\right| \left| ie^{i\theta}\, e^{iRt\sin\theta} \right| d\theta$$

By resolving $ie^{i\theta}\, e^{iRt\sin\theta}$ into the form $x + iy$ $(x,y \in \mathbb{R})$ it is easily shown that $|ie^{i\theta}\, e^{iRt\sin\theta}| \leqslant 1$. Hence

$$\left|\lim_{R\to\infty} \int_{BC} \right| \leqslant \sum_{m=0}^{\infty} \lim_{R\to\infty} \int_{\theta_o}^{\pi} Re^{Rt\cos\theta} \left|\phi_m(Re^{i\theta})\right| d\theta \qquad - (A3.1.1)$$

Now

$$|\phi_m(Re^{i\theta})| = |A_0^{(2m)}| \; \frac{|Fek_{2m}(\xi,-k^2)|}{|Fek_{2m}(\xi_o,-k^2)|} \; \left|\frac{ce_{2m}(\eta,-k^2)}{Re^{i\theta}}\right|$$

$$\leqslant |A_0^{(2m)}| \; \frac{|Fek_{2m}(\xi,-k^2)|}{|Fek_{2m}(\xi_o,-k^2)|} \; \frac{|ce_{2m}(\eta,-k^2)|}{R}$$

Since $\xi_o \leqslant \xi$ then $|Fek_{2m}(\xi,-k^2)| \leqslant |Fek_{2m}(\xi_o,-k^2)|$. Also, by the normalisation of the co-efficients $A_0^{(2m)}$, [(2), §2.21], $|A_0^{(2m)}| < 1$. Finally, the function ce_{2m} is a bounded function and so $|ce_{2m}(\eta,-k^2)| \leqslant M_m$, a constant for each m. Thus

$$|\phi_m(Re^{i\theta})| \leqslant \frac{M_m}{R} \qquad - (A3.1.2)$$

From (A3.1.1) and (A3.1.2)

$$\left|\lim_{R\to\infty} \int_{BC} \right| \leqslant \sum_{m=0}^{\infty} \lim_{R\to\infty} \int_{\theta_o}^{\pi} M_m\, e^{Rt\cos\theta}\, d\theta \qquad - (A3.1.3)$$

$$= \sum_{m=0}^{\infty} \lim_{R\to\infty} \left[\int_{\theta_o}^{\frac{\pi}{2}} M_m\, e^{Rt\cos\theta}\, d\theta + \int_{\frac{\pi}{2}}^{\pi} M_m\, e^{Rt\cos\theta}\, d\theta \right]$$

On the interval $\theta_o \leqslant \theta \leqslant \frac{\pi}{2}$, $\cos\theta \leqslant \cos\theta_o$ and the first integral of (A3.1.3) is less than or equal to

$$\sum_{m=0}^{\infty} M_m \lim_{R\to\infty} \left| e^{Rt\cos\theta_o} \left(\frac{\pi}{2} - \theta_o \right) \right|$$

$$= \sum_{m=0}^{\infty} M_m \lim_{R\to\infty} e^{\frac{Rta}{(a^2+R^2)^{\frac{1}{2}}}} \left(\frac{\pi}{2} - \cos^{-1}[a/(a^2+R^2)^{\frac{1}{2}}] \right)$$

(since for fixed 'a', $\cos\theta_o = a(a^2+R^2)^{-\frac{1}{2}}$) which is easily seen to be zero for each M_m.

On the interval $\frac{\pi}{2} \leqslant \theta \leqslant \pi$, $\cos\theta \leqslant 1 - \frac{2\theta}{\pi}$ and the second integral of (A3.1.3) is less than or equal to

$$\sum_{m=0}^{\infty} \lim_{R\to\infty} \int_{\frac{\pi}{2}}^{\pi} e^{Rt\left(1-\frac{2\theta}{\pi}\right)} \, d\theta$$

$$= -\sum_{m=0}^{\infty} M_m \lim_{R\to\infty} \frac{\pi}{2Rt} \left(e^{-Rt} - 1 \right)$$

which is zero for each M_m. Thus we have shown that the contour integral over the arc BC is zero in the limit $R \to \infty$. A similar argument to the above yields the same result for the arc FA as required.

(ii) Line Segments CD and EF

From [(7), § 8.30] we have

$$Fek_{2m}(\xi,-k^2) = (-1)^m \, Fek_{2m}\left(\frac{\pi i}{2}+\xi,+k^2\right)$$

$$= (-1)^m \frac{ce_{2m}(\frac{\pi}{2},k^2)}{\pi A_0^{(2m)}} \sum_{r=0}^{\infty} A_{2r}^{(2m)} K_{2r}[-2ik\cosh(\frac{\pi i+\xi}{2})]$$

- (A3.1.4)

on using [(5), § 8.14]. $K_{2r}(z)$ is a modified Bessel function. On the line segment CD, with reference to figure 3.2, we let $s = \rho e^{i\pi}$ and so $k^2 = \frac{h^2\rho e^{i\pi}}{4\kappa}$. (A3.1.4) yields

$$Fek_{2m}(\xi,-k^2) = (-1)^m \frac{ce_{2m}(\frac{\pi}{2},-\frac{h^2\rho}{4\kappa})}{\pi A_0^{(2m)}} \sum_{r=0}^{\infty} A_{2r}^{(2m)} K_{2r}[-2i(\frac{h^2\rho}{4\kappa})^{\frac{1}{2}} e^{i\pi}\cosh\xi]$$

Writing $w = \frac{h^2\rho}{4\kappa}$, noting that $ie^{\frac{i\pi}{2}} = -1$ and using [(1), § 8.40] yields

$$Fek_{2m}(\xi,-k^2) = \frac{ce_{2m}(0,\omega)}{\pi A_0^{(2m)}} \sum_{r=0}^{\infty} A_{2r}^{(2m)} K_{2r}(2\omega^{\frac{1}{2}} e^{\frac{i\pi}{2}}\cosh\xi)$$

- (A3.1.5)

From Abramowitz and Stegun (1964) [9.1.4] and [9.6.4] we have

$$H_\nu^{(2)}(z) = J_\nu(z) - iY_\nu(z)$$

$$K_\nu(z) = -\frac{i\pi}{2} e^{\frac{-\nu\pi i}{2}} H_\nu^{(2)}(ze^{\frac{-i\pi}{2}})$$

for $-\frac{\pi}{2} < \arg(z) \leqslant \pi$ and obtain

$$K_{2r}(ze^{\frac{i\pi}{2}}) = -\frac{i\pi}{2} e^{-r\pi i} \left(J_{2r}(z) - iY_{2r}(z) \right)$$

From (A3.1.5)

$$Fek_{2m}(\xi,-k^2)$$

$$= -i \frac{ce_{2m}(0,\omega)}{2 A_0^{(2m)}} \sum_{r=0}^{\infty} A_{2r}^{(2m)} (-1)^r \left[J_{2r}(2\omega^{\frac{1}{2}}\cosh\xi) - iY_{2r}(2\omega^{\frac{1}{2}}\cosh\xi) \right]$$

$$= - \frac{ce_{2m}(0,\omega)}{2\, ce_{2m}(\frac{\pi}{2},\omega)} \left[iCe_{2m}(\xi,\omega) + Fey_{2m}(\xi,\omega) \right]$$

on using [(15), § 8.10] and [(1), § 8.11]. Similarly for ξ_o we obtain

$$Fek_{2m}(\xi_o,k^2) = - \frac{ce_{2m}(0,\omega)}{2\, ce_{2m}(\frac{\pi}{2},\omega)} \left[iCe_{2m}(\xi_o,\omega) + Fey_{2m}(\xi_o,\omega) \right]$$

With $ce_{2m}(\Lambda,-k^2) \equiv ce_{2m}(\Lambda,\omega)$ and from Abramowitz and Stegun (1964), [20.8.5],

$$A_0^{(2m)}(k^2) \equiv A_0^{(2m)}(-\omega) = (-1)^m A_0^{(2m)}(\omega)$$

we obtain for the integral over the line segment CD, writing $\kappa = \frac{4k}{h^2}$,

$$\sum_{m=0}^{\infty} \int_{\infty}^{0} e^{-q\omega t} A_0^{(2m)} \frac{\left[iCe_{2m}(\xi,\omega) + Fey_{2m}(\xi,\omega) \right]}{\left[iCe_{2m}(\xi_o,\omega) + Fey_{2m}(\xi_o,\omega) \right]} ce_{2m}(\Lambda,\omega) \frac{d\omega}{\omega}$$

- (A3.1.6)

which on reversing the limits of integration and re-arranging the complex quotient yields the result (6) in chapter 3 section 1.

By a similar argument to that above using: $s = \rho e^{-i\pi}$ noting that [Abramowitz and Stegun (1964) [9.1.3] and [9.6.4]]

$$K_{2r}(ze^{\frac{-i\pi}{2}}) = \frac{\pi i}{2} e^{-r\pi i} [J_{2r}(z) + iY_{2r}(z)] , \quad -\pi < \arg(z) \leq \frac{\pi}{2}$$

we obtain the following for the integral over the line segment EF

$$\sum_{m=0}^{\infty} \int_{0}^{\infty} e^{-\kappa\omega t} A_0^{(2m)} [\frac{iCe_{2m}(\xi,\omega) - Fey_{2m}(\xi,\omega)}{iCe_{2m}(\xi_o,\omega) - Fey_{2m}(\xi_o,\omega)}] ce_{2m}(\eta,\omega) \frac{d\omega}{\omega}$$

- (A3.1.7)

It is easily shown that this is the negative complex conjugate of (A3.1.6) as required. Adding the two equations yields the required result (7) in chapter 3 section 1.

(iii) Circle DE (as the radius tends to zero)

For the small circle DE we set $s = \epsilon e^{i\phi}$ and consider the resulting integral as the radius ϵ tends to zero. Noting the direction of the contour, the integration is performed from $\phi = \pi$ to $\phi = -\pi$. This yields for the integral over DE

$$\lim_{\epsilon \to 0} \sum_{m=0}^{\infty} (-1)^m \int_{\pi}^{-\pi} e^{\epsilon e^{i\phi} t} A_0^{(2m)} \frac{Fek_{2m}(\xi,-\epsilon e^{i\phi})}{Fek_{2m}(\xi_o,-\epsilon e^{i\phi})} ce_{2m}(\eta,-\epsilon e^{i\phi}) \, id\phi$$

- (A3.1.8)

Now, $\lim_{\varepsilon \to 0} e^{\varepsilon e^{i\phi} t} = 1$ for all t and $\phi \in (-\pi, \pi)$. Also, [§ 3.32], $\lim_{\varepsilon \to 0} A_0^{(2m)}(\varepsilon) = 0$ for $m \neq 0$. Due to the normalisation of Mathieu functions we have, [§ 2.21], $\lim_{\varepsilon \to 0} A_0^{(0)}(\varepsilon) = \frac{1}{\sqrt{2}}$. From [(2), § 2.18]

$$\lim_{\varepsilon \to 0} ce_{2m}(\eta, -\varepsilon e^{i\phi}) = \lim_{\varepsilon \to 0} \sum_{r=0}^{\infty} A_{2r}^{(0)}(\omega) \cos 2r\eta = \frac{1}{\sqrt{2}}$$

From [(8), § 8.30]

$$Fek_0(\xi, -\varepsilon e^{i\phi}) \simeq K_0(2\varepsilon^{\frac{1}{2}} e^{\frac{i\phi}{2}} \sinh\xi)$$

for $\varepsilon \simeq 0$. Hence (A3.1.8) becomes

$$\lim_{\varepsilon \to 0} \frac{i}{2} \int_{\pi}^{-\pi} \frac{K_0(2\varepsilon^{\frac{1}{2}} e^{\frac{i\phi}{2}} \sinh\xi)}{K_0(2\varepsilon^{\frac{1}{2}} e^{\frac{i\phi}{2}} \sinh\xi_0)} \, d\phi = -\pi i \qquad \text{- (A3.1.9)}$$

as required.

APPENDIX 4.1

VAX 11/780 FORTRAN Code for the Sensitivity Analysis of Section 4.3

VARIABLES :	var(8)	Sensitivity variables (as shown in the DATA statement for par(8))
	dk(2,3)	1st derivitive of [1] solid & [2] liquid conductivities w.r.t. {[1] wall, [2] fusion, [3]liquid} temperature
	ds	Solid diffusivity
	dxo	Sensitivity of the freezing front w.r.t. the chosen parameter
	dlam	As 'dxo' for the freezing paramter lambda
	time	Maximum time for the simulation
	ipar	Parameter number
	ivar	Number of evaluations
	varist	1st parameter value
	varinc	Parameter increment
	ivol	0 for NO volume change, otherwise v. change

```
      program sensitive

*************************************************************************
* PROGRAM : To execute a sensitivity analysis for a 'real variables' *
*           one-dimensional  freezing problem. The  location  of the *
*           freezing front is determined as a function of one of the *
*           available  parameters. The  choice  of accounting  for a *
*           volume change on solidification also exists.             *
*************************************************************************

      character*20 par(8)
      character*1  in

      real*8       var(8),time,ds,lam,dlam,dxo,dk(2,3)

      data         par/'Init. Liqu. Temp. > ','Fusion Temp.      > ',
     c                 'Wall Temp.        > ','Solid Density     > ',
     c                 'Liquid Density    > ','Latent Ht. Fusion > ',
     c                 'Spec. Heat        > ','Lambda            > '/
```

```
+-----------------------------------------------------------------------+
| SELECT TEST PARAMETER                                                 |
+-----------------------------------------------------------------------+
```

```
      write(6,100)
    1 read(5,*)ipar
      if( ipar.lt.1.or.ipar.gt.8 ) then
          write(6,101)
          go to 1
      end if
```

```
+-----------------------------------------------------------------------+
| INPUT 'NON' TEST PARAMETER VALUES                                     |
+-----------------------------------------------------------------------+
```

```
      do 3 i = 1,7
      if( i.ne.ipar ) then
           write(6,102)par(i)
           read(5,*)var(i)
      end if
    3 continue
```

```
| INPUT TEST PARAMETER DATA                                                |
```

```
      type *,' '
      write(6,103)
      read(5,*)varist
      write(6,104)
      read(5,*)ivar
      write(6,105)
      read(5,*)varinc
```

```
| INPUT REAL TIME OUTPUT & SELECT VOLUME/NO VOLUME CHANGE                  |
```

```
      write(6,106)
      read(5,*)time
      write(6,107)
      read(5,*)ivol
```

```
| ENTER CALCULATION LOOP :  INCREMENT TEST PARAMETER                       |
```

```
      do 2 i = 1,ivar
      var(ipar) = varist + (i-1)*varinc
```

```
| CALCULATE SOLID DIFFUSIVITY IF IPAR = 8 (LAMBDA)                         |
```

```
      if(ipar.eq.8) then
           call conduct( var(2),var(3),ks )
           ds = ks/(var(4)*var(7))
           lam = var(8)
           go to 10
      end if
```

```
| CALCULATE FIRST DERIVITIVE OF SOLID  & LIQUID  CONDUCTIVITIES WITH       |
| RESPECT TO WALL,FUSION AND LIQUID TEMPERATURES                           |
```

```
      dk(1,1) = 0.0
      dk(1,2) = (0.96089239/((var(2)-var(3))*(var(2)-var(3))))
     c               *(0.00112*var(2)*var(2)
     c                +var(3)*(1.77+0.001*(0.2775*var(3)-2.24*var(2)))
     c               -877.5)
      dk(1,3) = (0.96089239/((var(2)-var(3))*(var(2)-var(3))))
```

```
c                *(0.0002775*var(3)*var(3)
c                +var(2)*(-1.77+0.001*(1.12*var(2)-0.455*var(3)))
c                +877.5)
      dk(2,1) = 0.00112*0.96089239
      dk(2,2) = 0.00112*0.96089239
      dk(2,3) = 0.0
```

```
| EVALUATE THE FREEZING PARAMETER LAMBDA                                |
```

```
      call lambda( lam,var,ipar,ds,dlam,ivol,dk,ks )
```

```
| EVALUATE THE FREEZING FRONT LOCATION AT 'time'                        |
```

```
      var(8) = lam
   10 bound = 2.0*lam*dsqrt(ds*time)
```

```
| EVALUATE THE  FIRST DERIVITIVE OF THE  FREEZING FRONT WITH RESPECT    |
| TO THE TEST PARAMETER                                                 |
```

```
      go to (11,12,13,14,15,16,17,18),ipar
   11 dxo = (dlam*bound)/lam
      go to 19
   12 dxo = (bound/(2.0*lam))*(2.0*dlam + (lam*dk(1,2))/ks)
      go to 19
   13 dxo = (bound/(2.0*lam))*(2.0*dlam + (lam*dk(1,3))/ks)
      go to 19
   14 dxo = (bound/(2.0*lam))*(2.0*dlam - lam/var(4))
      go to 19
   15 dxo = (dlam*bound)/lam
      go to 19
   16 dxo = (dlam*bound)/lam
      go to 19
   17 dxo = (bound/(2.0*lam))*(2.0*dlam - lam/var(7))
      go to 19
   18 dxo = bound/lam
      dlam = 1.0

   19 write(1,108)var(ipar),1000.0*bound,lam,dlam,dxo

    2 continue
```

```
| FORMAT STATEMENTS                                                     |
```

```
  100 format(1x,'Variable to be tested'/1x/
     c       1x,'1) Init. Liqu. Temp.'/
     c       1x,'2) Fusion Temp.     '/
     c       1x,'3) Wall Temp.       '/
     c       1x,'4) Solid Density    '/
     c       1x,'5) Liquid Density   '/
     c       1x,'6) Latent Ht. Fusion'/
```

```
     c        1x,'7) Specific Heat    '/
     c        1x,'8) Lambda                > ',$)
  101  format(1x,'What ?                   > ',$)
  102  format(1x,A20,$)
  103  format(1x,'Initial variable value > ',$)
  104  format(1x,'No. of evaluations     > ',$)
  105  format(1x,'Increment of variable  > ',$)
  106  format(1x,'Time of interface loc. > ',$)
  107  format(1x,'Vol. change ? 0/1      > ',$)
  108  format(1x,2f16.8,3f15.10)

       stop
       end

       subroutine lambda( lam,var,ipar,ds,dlam,ivol,dk,ks )

*************************************************************************
* SUBROUTINE : This routine calculates the freezing parameter lambda *
*              (see equations (4a), no volume change, and (4b), *
*              volume change) using the bisection method. The first *
*              derivitive of lambda with respect to the test *
*              parameter [var(ipar)] is then evaluated.          *
*************************************************************************

       real*8 lam,to,tf,tw,pl,ps,c,dl,ds,dlam(2,3),
     c        ll,l2,lat,rpi,s15aef,s15adf,s1,s2,ff,dd,ss,kl,ks,
     c        f1,f2,f3,f4,f5,f6,f7,f8,df(8,7,2),cof(2),var(8)

       ifail = 0
       rpi   = 2.0d0*dsqrt(datan(1.0d0))
       ll    = 0.0
       l2    = 2.0
       to    = var(1)
       tf    = var(2)
       tw    = var(3)
       ps    = var(4)
       pl    = var(5)
       lat   = var(6)
       c     = var(7)

+-----------------------------------------------------------------------+
| CALCULATE SOLID & LIQUID CONDUCTIVITIES AND DIFFUSIVITIES             |
+-----------------------------------------------------------------------+

       call conduct( tf,tw,ks )
       call conduct( to,tf,kl )

       dl = kl/(pl*c)
       ds = ks/(ps*c)

+-----------------------------------------------------------------------+
| EVALUATE LAMBDA. f1,.,f8 ARE FUNCTIONS WHICH MAKE UP EQUATION (4a)    |
| (NO VOLUME CHANGE) OR EQUATION (4b) (VOLUME CHANGE)                   |
+-----------------------------------------------------------------------+

       f3 = (to-tf)/(tf-tw)
       f4 = dsqrt(kl/ks)
       f5 = dsqrt(pl/ps)
```

```
10   lam = (l1+l2)/2.0d0

     f1 = dexp(-lam*lam)
     f2 = s15aef(lam,ifail)
     if( ivol.eq.0 ) then
          f6 = dexp(-(lam*lam*f5*f5)/(f4*f4))
          f7 = s15adf((lam*f5)/f4,ifail)
          else
          f6 = dexp(-(lam*lam)/(f4*f4*f5*f5))
          f7 = s15adf(lam/(f4*f5),ifail)
     end if
     f8 = (lat*lam*rpi)/(c*(tf-tw))

     ff = (f1/f2) - (f3*f4*f5*f6)/f7 - f8

     if( dabs(ff).le.1.0d-12 ) go to 20
     if( ff.lt.0.0 ) then
          l2 = lam
          else
          l1 = lam
     end if
     go to 10
20   continue
     if( ipar.eq.8 ) return
```

```
+-----------------------------------------------------------------------+
| EVALUATE D(LAMBDA)/D(VAR(IPAR)) :                                     |
| CALCULATE THE FIRST DERIVITIVES  OF THE ABOVE 8 'F' FUNCTIONS WITH    |
| RESPECT TO THE  RELEVENT TEST PARAMETER  VAR(IPAR) AND STORE   THE    |
| CONSTANT COEFFICIENTS AND THE COEFFICIENTS OF DLAMBDA                 |
| DF(I,J,K) = DERIVITIVE OF THE Ith FUNCTION WITH RESPECT TO THE Jth    |
|             PARAMETER. K = (1,2) => (CONSTANT,DLAMBDA) COEFFICIENT    |
+-----------------------------------------------------------------------+
```

```
+-----------------------------------------------------------------------+
| DERIVITIVES OF F1                                                     |
+-----------------------------------------------------------------------+
```

```
     do 1 j = 1,7
     df(1,j,1) = 0.0
 1   df(1,j,2) = -2.0*lam*f1
```

```
+-----------------------------------------------------------------------+
| .. OF F2                                                              |
+-----------------------------------------------------------------------+
```

```
     do 2 j = 1,7
     df(2,j,1) = 0.0
 2   df(2,j,2) = (2.0*f1)/rpi
```

```
+-----------------------------------------------------------------------+
| .. OF F3                                                              |
+-----------------------------------------------------------------------+
```

```
     df(3,1,1) = 1.0/(tf-tw)
     df(3,2,1) = (tw-to)/((tf-tw)*(tf-tw))
     df(3,3,1) = f3/(tf-tw)
```

```
      do 3 j = 4,7
    3 df(3,j,1) = 0.0
      do 13 j = 1,7
   13 df(3,j,2) = 0.0
```

```
| .. OF F4                                                                |
```

```
      df(4,1,1) = dk(2,1)/(2.0*f4*ks)
      df(4,2,1) = (0.5/ks)*(dk(2,2)/f4 - f4*dk(1,2))
      df(4,3,1) = -(f4*dk(1,3))/(2.0*ks)
      do 4 j = 4,7
    4 df(4,j,1) = 0.0
      do 14 j = 1,7
   14 df(4,j,2) = 0.0
```

```
| .. OF F5                                                                |
```

```
      do 5 j = 1,3
    5 df(5,j,1) = 0.0
      df(5,4,1) = -f5/(2.0*ps)
      df(5,5,1) =  f5/(2.0*p1)
      df(5,6,1) = 0.0
      df(5,7,1) = 0.0
      do 15 j = 1,7
   15 df(5,j,2) = 0.0
```

```
| .. OF F6                                                                |
```

```
      do 6 j = 1,3
      if( ivol.eq.0 ) then
           df(6,j,1) = (2.0*lam*lam*f6*f5*f5*df(4,j,1))/(f4*f4*f4)
           else
           df(6,j,1) = (2.0*lam*lam*f6*df(4,j,1))/(f4*f4*f4*f5*f5)
      end if
    6 continue
      do 16 j = 4,5
      if( ivol.eq.0 ) then
           df(6,j,1) = -(2.0*lam*lam*f6*f5*f5*df(5,j,1))/(f4*f4*f5)
           else
           df(6,j,1) = -(2.0*lam*lam*f6*df(5,j,1))/(f4*f4*f5*f5*f5)
      end if
   16 continue
      do 26 j = 6,7
   26 df(6,j,1) = 0.0
      do 36 j = 1,7
      if( ivol.eq.0 ) then
           df(6,j,2) = -(2.0*lam*f6*f5*f5)/(f4*f4)
           else
           df(6,j,2) = -(2.0*lam*f6)/(f4*f4*f5*f5)
      end if
   36 continue
```

```
| .. OF F7                                                                |
```

```
      do 7 j = 1,3
      if( ivol.eq.0 ) then
           df(7,j,1) = (2.0*lam*f5*f6*df(4,j,1))/(rpi*f4*f4)
           else
           df(7,j,1) = (2.0*lam*f6*df(4,j,1))/(rpi*f4*f4*f5)
      end if
    7 continue
      do 17 j = 4,5
      if( ivol.eq.0 ) then
           df(7,j,1) = -(2.0*lam*f6*df(5,j,1))/(rpi*f4)
           else
           df(7,j,1) = -(2.0*lam*f6*df(5,j,1))/(rpi*f4*f4*f5)
      end if
   17 continue
      do 27 j = 6,7
   27 df(7,j,1) = 0.0
      do 37 j = 1,7
      if( ivol.eq.0 ) then
           df(7,j,2) = -(2.0*f5*f6)/(rpi*f4)
           else
           df(7,j,2) = -(2.0*f6)/(rpi*f4*f5)
      end if
   37 continue
```

```
| .. OF F8                                                                |
```

```
      df(8,1,1) = 0.0
      df(8,2,1) = -f8/(tf-tw)
      df(8,3,1) =  f8/(tf-tw)
      df(8,4,1) = 0.0
      df(8,5,1) = 0.0
      df(8,6,1) = f8/lat
      df(8,7,1) = -f8/c
      do 8 j = 1,7
    8 df(8,j,2) = f8/lam
```

```
| EVALUATE  COMPLETE  COEFFICIENTS OF CONSTANTS (COF(1)) AND DLAMBDA      |
| (COF(2))                                                                |
```

```
      do 9 m = 1,2
    9 cof(m) = f7*f7*(f2*df(1,ipar,m) - f1*df(2,ipar,m))
     c        - f2*f2*(f7*(f3*f4*f5*df(6,ipar,m)
     c        +            f3*f4*f6*df(5,ipar,m)
     c        +            f3*f5*f6*df(4,ipar,m)
     c        +            f4*f5*f6*df(3,ipar,m))
     c        -         f3*f4*f5*f6*df(7,ipar,m))
     c        - f2*f2*f7*f7*df(8,ipar,m)
```

```
+-----------------------------------------------------------------------+
| DETERMINE DLAMBDA/DVAR(IPAR)                                          |
+-----------------------------------------------------------------------+

      dlam = -cof(1)/cof(2)

      return
      end

      subroutine conduct( tup,tlo,cond )

*************************************************************************
* SUBROUTINE : This routine calculates the average conductivity over *
*              a temperature range using equations (4c), section 4.3 *
*              and Carslaw & Jaeger (1959), p.93                      *
*************************************************************************

      real*8 tup,tlo,cond

      cond = 0.0

      if( tup.le.1000.0.and.tlo.le.tup ) then
           cond = 0.65*0.96089239*(1.0 + 0.00035*(tup+tlo))
           else
           if( tup.gt.1000.0.and.tlo.le.1000.0 ) then
                cond = (0.96089239/(tup-tlo))
     c                *(877.5
     c                  -0.65*tlo*(1.0 + 0.00035*tlo)
     c                  -1.12*tup*(1.0 - 0.00100*tup))
                else
                if( tlo.gt.1000.0.and.tup.ge.tlo ) then
                     cond = 1.12*0.96089239*(-1.0 + 0.001*(tup+tlo))
                end if
           end if
      end if

      return
      end
```

APPENDIX 4.2

Finite Difference Equations for Equations (46) and (37) of Chapter IV

We first examine the form of equation (46), which governs the liquid temperature distribution, as $\lambda \rightarrow 0$. Included in the expression is a term $K_L/\lambda H^2 \frac{\partial U_L}{\partial \lambda}$. By the symmetry of the problem (about the z-axis) we know that $\frac{\partial U_L}{\partial \lambda} = 0.0$ at $\lambda = 0$ and so the above term tends to $\frac{0}{0}$ (as $\lambda \rightarrow 0$) which is an indeterminate value. To evaluate the term we consider the following Maclaurins expansion.

$$\frac{\partial U_L}{\partial \lambda}(\lambda) = \frac{\partial U_L}{\partial \lambda}(0) + \lambda \frac{\partial^2 U_L}{\partial \lambda^2}(0) + \frac{\lambda^2}{2} \frac{\partial^3 U_L}{\partial \lambda^3}(0) + \ldots$$

Since $\frac{\partial U_L}{\partial \lambda}(0) = 0$ then

$$\frac{\partial U_L}{\partial \lambda}(\lambda) = \lambda \frac{\partial^2 U_L}{\partial \lambda^2}(0) + \frac{\lambda^2}{2} \frac{\partial^3 U_L}{\partial \lambda^3} + O(\lambda^3)$$

Hence,

$$\frac{1}{\lambda} \frac{\partial U_L}{\partial \lambda}(\lambda) = \frac{\partial^2 U_L}{\partial \lambda^2}(0) + \frac{\lambda}{2} \frac{\partial^3 U_L}{\partial \lambda^3}(0) + O(\lambda^2)$$

and thus

$$\lim_{\lambda \rightarrow 0} K_L/\lambda H^2 \frac{\partial U_L}{\partial \lambda} = K_L/H^2 \frac{\partial^2 U_L}{\partial \lambda^2}(0)$$

At $\lambda = 0$ equation (46) then becomes

$$Q[1-\frac{1}{2}\sin(\pi\omega t)] \frac{\partial U_L}{\partial z} = K_L \frac{\partial^2 U}{\partial \lambda^2} \qquad - (A4.2.1)$$

for $\lambda = 0$, $0 < z \leqslant D$, $t > 0$. Using central differences for the spacial derivitives, the explicit difference approximation to equations (46) (chapter IV) and (A4.2.1) are

$$\frac{2Q(1-\lambda_i^2)}{K_L}\left[1-\frac{1}{2}\sin(\pi\omega t)\right]\frac{\partial U_L}{\partial z} =$$

$$+ 2\left\{\frac{(\Delta\lambda_{i+1}U_{i-1,j} + \Delta\lambda_i U_{i+1,j})}{(\Delta\lambda_i + \Delta\lambda_{i+1})} - U_{i,j}\right\}\frac{1}{\Delta\lambda_i\Delta\lambda_{i+1}}$$

$$+ \frac{(U_{i+1,j} - U_{i-1,j})}{(\Delta\lambda_i + \Delta\lambda_{i+1})}$$

for $1 \leqslant i \leqslant N_\lambda - 1$, $1 \leqslant j \leqslant N_z$, $m \geqslant 0$ and

$$Q\left[1-\frac{1}{2}\sin(\pi\omega t)\right]\frac{\partial U_L}{\partial t} = K_L\frac{(U_{1,j} - U_{0,j})}{(\Delta\lambda_i + \Delta\lambda_{i+1})}$$

for $i = 0$, $1 \leqslant j \leqslant N_z$, $m \geqslant 0$. The following standard notation is used in the above expressions. $U_{i,j} \simeq U_L(\lambda_i, j\Delta z)$ is the temperature at the node $(\lambda_i, j\Delta z, t^m)$ where t^m is the cumulative time after m time steps. λ_i is the i^{th} radial node and $\Delta\lambda_i$ is the length of the i^{th} radial mesh segment. The non-standard radial differences in the above expressions arise because of the non-uniform radial grid that is chosen for the problem.

With regard to the Runga-Kutta solution for these equations we take the right-hand side as a function f of z, t and $U_{i,j-1}$. The following four values k_1, k_2, k_3 and k_4 are then calculated.

$$k_1 = f(\ t,\ z\ ,U_{i,j-1}\ )$$
$$k_2 = f(\ t,\ z+0.5*dz,U_{i,j-1}+0.5*dz*k_1\ )$$
$$k_3 = f(\ t,\ z+0.5*dz,U_{i,j-1}+0.5*dz*k_2\ )$$
$$k_4 = f(\ t,\ z+dz\ ,U_{i,j-1}+dz*k_3\ )$$

The value $U_{i,j}$ is then calculated from the expression

$$U_{i,j} = U_{i,j-1} + \{\ dz\ (\ k_1 + 2k_2 + 2k_3 + k_4\ )\ \}/6$$

For the interface (equation (37), chapter IV) we first calculate the velocity $\partial H/\partial t$ at the m^{th} time step using the following explicit difference approximation

$$\frac{\partial H}{\partial t} = \frac{1}{lp}\left\{\frac{kts}{H_j^m \ln(R_j/H_j^m)} - \frac{ktl}{H_j^m}\nabla_j^m + qc\right\} \qquad - (A4.2.2)$$

where qc is the convective heat flux (at the interface) $h(T_f-T_w)$ and ∇_j^m is the 'radial' liquid temperature gradient at the interface, at the j^{th} axial node and m^{th} time step. The remaining constants in the above expression are given as follows.

$$kts = k_s(T_w-T_f)$$
$$ktl = k_L(T_o-T_f)$$
$$lp = L\rho_S$$

Using a backward difference approximation for the time derivitive $\partial H/\partial t$, the new interface location at the j^{th} axial node is given by (with A4.2.2)

$$H_j^{m+1} = H_j^m + \Delta t \frac{\partial H}{\partial t}, \quad 1 \leqslant j \leqslant N_z, \; m \geqslant 0$$

APPENDIX 4.3

VAX 11/780 FORTRAN Code for the Industrial Freezing Problem

The FORTRAN code listed contains local non-standard characteristics. Several possibilities exist for using the program. The item 'plotno' determines the type of simulation. A value from 1 to 10 will produce results giving the crust thickness at some specified point 'dout' on the wall as a function of one of the physical parameters (see PRELDATA format statements 101 to 500 for these). 11 gives the crust as a function of time and 12 gives the crust profile as a function of time. CHILLGAS is the driving program. Data is read in from a file on channel 012. Interface output is written to observable data files (on channels 041 to 052 depending on 'plotno') and to files that may be used as a database for plotting (channels 061 to 072). If temperature calculations are required the routine PROGKUTT replaces PROGFACE and data is written on channel 099. A summary of the routines follows with a description of the main variables used.

ROUTINES :	CHILLGAS	Driving program
	CRUSDATA	Writes output for 'plotno' from 1 to 11
	INTERPOL	Estimates normal crust thickness at point 'dout'
	LAMBDA	Calculates 1-D freezing parameter 'lambda'
	ERF	Function that returns the error function
	PRELDATA	Writes preliminary data to relevant files
	PROFDATA	Writes output for 'plotno' 12
	PROGFACE	Calculates interface at successive times
	RADMESHS	Defines radial mesh configuration
	READDATA	Reads in data for simulation model
	STARTERS	Initialises the model
	VARCONST	Defines 'variable' constants
	VOLUFLUX	Evaluates the volume flux
	PROGKUTT	Replaces PROGFACE when temperature calculations are required.

VARIABLES :	u(0:28,0:25)	Discrete temperature distribution
	h(0:25)	Discrete interface location
	l(0:28),dl(28)	Radial nodes & mesh sizes
	axipts,dz	Number & size of axial nodes/segments
	tmax,t,dt	Maximum, current & incremental times
	vflux	Volume flux
	var(10)	Ten physical parameters
	plotno	Type of simulation (see above)
	varinc,varnum	Increment & number of var(plotno) calcs.

The remaining variables are fairly clear from the program or the text.

```
      program chillgas

*********************************************************************
* PROGRAM : This program computes a  finite-difference solution to a *
*           Stefan Problem connected with the solidification of slag *
*           in a slagging gasifier. The calculations are  real time. *
*********************************************************************

      logical   steady,blocked,more_data

      integer*4 plotno,varnum

      real*8    t,tmax

      common    /logi1/steady,blocked
      common    /misc1/more_data
      common    /time1/tmax/time4/t
      common    /vars2/plotno/vars4/varnum

    1 call readdata
      call radmeshs
      call preldata
      do nv = 1,varnum
           call varconst
           call starters
           do while( t.lt.tmax.and..not.steady.and..not.blocked )
                call voluflux
                call progress
                if( plotno.eq.12 ) call profdata
                if( plotno.le.11 ) call crusdata
           end do
           blocked = .false.
           steady  = .false.
      end do

      if( more_data ) go to 1

      stop
      end

      subroutine crusdata

*********************************************************************
* SUBROUTINE : This  routine writes  data calculated  by plot numbers *
*              1 through 11 to the relevant  output  and plot  files. *
*********************************************************************

      logical   steady,blocked

      integer*4 plotno,tout,nt

      real*8    t,tmax,var(10),x,y,dout,pi

      common    /cons4/pi
```

```
      common    /dout1/dout
      common    /logi1/steady,blocked
      common    /time1/tmax/time2/tout/time4/t/time5/nt
      common    /vars1/var/vars2/plotno

      if( t.eq.tmax.or.steady.or.blocked ) then
           go to 10
           else
           if( plotno.eq.11.and.mod(nt,tout).eq.0 ) go to 10
      end if

      return

 10   if( blocked ) then
           y = ( (var(8)+dout*dcos(var(10)*pi))*1000.0 )
     c        /dsin(var(10)*pi)
           else
           call interpol( y )
      end if

      go to( 1,2 ),jifix( plotno/11.0 ) + 1

  1   x = var(plotno)
      if( .not.blocked.and..not.steady ) then
           write(plotno+40,100)x,y,t
           write(plotno+60,103)x,y,t,0
           else
           if( steady ) then
                write(plotno+40,102)x,y,t
                write(plotno+60,103)x,y,t,1
                else
                write(plotno+40,101)x,y,t
                write(plotno+60,103)x,y,t,2
           end if
      end if

      return

  2   if( .not.blocked.and..not.steady ) then
           write(51,100)t,y
           write(71,104)t,y,0
           else
           if( steady ) then
                write(51,105)t,y
                write(71,104)t,y,1
                else
                write(51,106)t,y
                write(71,104)t,y,2
           end if
      end if

      return

100   format(1x,f12.4,31x,f16.8,10x,f10.4)
101   format(1x,f12.4,31x,f16.8,10x,'BLOCKAGE for time <=',f10.4,
     c                              ' seconds')
102   format(1x,f12.4,31x,f16.8,10x,'STEADY-STATE at time =',f10.4,
     c                              ' seconds')
```

```
  103  format(1x,3f16.8,i3)
  104  format(1x,2f16.8,i3)
  105  format(1x,f12.4,31x,f16.8,10x,' - STEADY-STATE')
  106  format(1x,f12.4,31x,f16.8,10x,' - BLOCKAGE')

       end

       subroutine interpol(c)

*************************************************************************
* SUBROUTINE : This routine estimates the 'normal' crust thickness in *
*              millimetres for the point 'dout' along the hearth wall *
*              using linear interpolation.                            *
*************************************************************************

       integer*4 axipts

       real*8    c,cota,var(10),pi,dout,h(0:25),rn,zn,r1,r2,h1,h2,
     c           gi,ki,rc,zc,dz,z1,z2

       common    /cons4/pi
       common    /dout1/dout
       common    /face1/h
       common    /incr2/dz
       common    /pnts2/axipts
       common    /vars1/var

       if( var(10).eq.0.5d0 ) then
            nj = jifix( sngl(dout/dz) )
            if( dble(nj*dz).eq.dout ) then
                 h1 = h(nj)
                 else
                 h1 = ( h(nj) + h(nj+1) )/2.0
            end if
            c = ( var(8) - h1 )*1000.0

            return
       end if

       cota = dcos(var(10)*pi)/dsin(var(10)*pi)
       rn   = var(8) + dout*dcos(var(10)*pi)
       zn   = dout*dsin(var(10)*pi)

       do 1 j = 2,axipts

       z1 = dble( floatj(j-1) )*dz
       z2 = dble( floatj(j)   )*dz
       h1 = h(j-1)
       h2 = h(j)

       r1 = rn + (zn-z1)*datan( var(10)*pi )
       r2 = rn + (zn-z2)*datan( var(10)*pi )

       if( r1.ge.h1.and.r2.le.h2 ) go to 2

    1  continue

    2  gi = (z2-z1)/(h2-h1)
```

```
      ki = (z1*h2 - z2*h1)/(h2-h1)

      rc = (rn*cota + zn - ki)/(cota + gi)
      zc = gi*rc + ki

      c  = dsqrt( (rc-rn)*(rc-rn) + (zc-zn)*(zc-zn) )*1000.0

      return
      end

      subroutine lambda( lam )

*************************************************************************
* SUBROUTINE : This routine calculates the freezing parameter for the *
*              1-d approximation (starting solution).                   *
*************************************************************************

      real*8  rtpi,l1,l2,to,tf,tw,f1,f2,f3,d1,d2,d3,erfx,erfc,lam,l,
     c        var(10),denstl,densts,spehtl,spehts,thdifs,thdifl,
     c        thconl,thcons

      common  /phys2/thconl,thcons/phys3/thdifl,thdifs
      common  /phys4/denstl,densts/phys5/spehtl,spehts
      common  /varsl/var

      rtpi = 2.0d0*dsqrt(datan(1.0d0))
      to   = var(3)
      tf   = var(2)
      tw   = var(1)
      l    = var(4)

      l1 = 1.0d0

    1 erfx = erf( l1 )
      erfc = 1.0 - erf( l1*dsqrt(thdifs/thdifl) )

      f1 = ( (tf-tw)*dexp(-l1*l1) )/erfx

      f2 = ( (to-tf)*thconl*dsqrt(thdifs)*dexp(-(thdifs*l1*l1)/thdifl) )
     c    /( thcons*dsqrt(thdifl)*erfc )

      f3 = ( l1*l*rtpi )/spehts

      d1 = ( ( 2.0d0*dexp(-l1*l1)*(tf-tw) )/( rtpi*erfx*erfx ) )
     c    *( rtpi*l1*erfx + dexp(-l1*l1) )

      d2 = ( (2.0d0*(to-tf)*thconl*thdifs*dexp(-(thdifs*l1*l1)/thdifl) )
     c      /( rtpi*thcons*thdifl*thdifl*erfc*erfc) )
     c    *( rtpi*dsqrt(thdifs)*l1*erfc - dsqrt(thdifl)
     c       *dexp(-(thdifs*l1*l1)/thdifl))

      d3 = (l*rtpi)/spehts

      l2 = l1 - (f1 - f2 - f3)/(- d1 + d2 - d3)

      if( dabs(l2-l1).le.1.0d-08 ) go to 2

      l1 = l2
```

```
      go to 1

    2 lam = 12

      return
      end

      double precision function erf( x )

*************************************************************************
* FUNCTION : This function returns  the  value of the  error function *
*            erf(x) at the  point x. The  calculation is  made  using *
*            Simpsons Rule with 100 points.                           *
*************************************************************************

      real*8 x,piroot,f,h

      piroot = 2.0*dsqrt(datan(1.0d0))

      if( x.lt.0.0 ) x = - x

      h = x/100.0
      f = 1.0 - dexp( -x*x )

      do 1 i = 1,99,2
    1 f = f + 2.0*dexp(-i*i*h*h)*( 2.0+dexp(-(2.0*i+1.0)*h*h) )

      erf = (2.0*h*f)/(3.0*piroot)

      return
      end

      subroutine preldata

*************************************************************************
* SUBROUTINE : This  routine writes  preliminary  data  to the output *
*              files and plotting data files.                           *
*************************************************************************

      integer*4 axipts,plotno,varnum

      real*8    dout,var(10),tmax,denstl,densts,spehtl,spehts

      common    /dout1/dout
      common    /phys4/denstl,densts/phys5/spehtl,spehts
      common    /pnts2/axipts
      common    /time1/tmax
      common    /vars1/var/vars2/plotno/vars4/varnum

      if( plotno.le.11 ) then
           write(plotno+40,100)
           if( plotno.eq.1 ) write(41,101)
           if( plotno.eq.2 ) write(42,102)
           if( plotno.eq.3 ) write(43,103)
           if( plotno.eq.4 ) write(44,104)
           if( plotno.eq.5 ) write(45,105)
           if( plotno.eq.6 ) write(46,106)
           if( plotno.eq.7 ) write(47,107)
```

```
      if( plotno.eq.8 ) write(48,108)
      if( plotno.eq.9 ) write(49,109)
      if( plotno.eq.10 )write(50,500)
      if( plotno.eq.11 )write(51,601)
    end if

    if( plotno.eq.12 ) write(52,123)
    if( plotno.le.11 ) write(plotno+40,603)
    if( plotno.le.10 ) write(plotno+40,110)tmax

    write(plotno+40,604)denstl,spehtl,densts,spehts

    if( plotno.ne.1  )write(plotno+40,111)var(1)
    if( plotno.ne.2  )write(plotno+40,112)var(2)
    if( plotno.ne.3  )write(plotno+40,113)var(3)
    if( plotno.ne.4  )write(plotno+40,114)var(4)
    if( plotno.ne.5  )write(plotno+40,115)var(5)
    if( plotno.ne.6  )write(plotno+40,116)var(6)
    if( plotno.ne.7  )write(plotno+40,117)var(7)
    if( plotno.ne.8  )write(plotno+40,118)var(8)
    if( plotno.ne.9  )write(plotno+40,119)var(9)
    if( plotno.ne.10 )write(plotno+40,501)var(10)*180.0d0
    if( plotno.le.11 )write(plotno+40,502)dout

    write(plotno+40,120)axipts

    if( plotno.eq.1  ) write(41,101)
    if( plotno.eq.2  ) write(42,102)
    if( plotno.eq.3  ) write(43,103)
    if( plotno.eq.4  ) write(44,104)
    if( plotno.eq.5  ) write(45,105)
    if( plotno.eq.6  ) write(46,106)
    if( plotno.eq.7  ) write(47,107)
    if( plotno.eq.8  ) write(48,108)
    if( plotno.eq.9  ) write(49,109)
    if( plotno.eq.10 ) write(50,500)
    if( plotno.eq.11 ) write(51,601)

    if( plotno.le.11 ) write(plotno+40,602)

    write(plotno+60,200)varnum,tmax

100 format(1x,'CRUST THICKNESS (mm) AS A FUNCTION OF ',$)
101 format('+','WALL TEMPERATURE (Deg. C)              ',$)
102 format('+','FUSION TEMPERATURE (Deg. C)            ',$)
103 format('+','INITIAL LIQUID TEMPERATURE (Deg. C)    ',$)
104 format('+','LATENT HEAT OF FUSION (J/kg)           ',$)
105 format('+','CONVECTIVE HEAT CO-EFFICIENT (J/m s C)',$)
106 format('+','INITIAL BULK SLAG VELOCITY (m/s)       ',$)
107 format('+','BUBBLE FREQUENCY (/s)                  ',$)
108 format('+','TUBE RADIUS (m)                        ',$)
109 format('+','TUBE LENGTH (Slag Depth) (m)           ',$)
500 format('+','WALL ELEVATION (degrees)               ',$)
601 format('+','TIME (s)                               ',$)
602 format('+',5x,'CRUST THICKNESS (mm)')
603 format(1x)
110 format(1x/1x,'TIME =',f7.2,' s')
604 format(1x/1x,'PHYSICAL DATA'/1x/
```

```
     c          1x,'Liquid Density              :',f10.4,' kg/m '/
     c          1x,'Liquid Specific Heat        :',f10.4,' J/kg C'/
     c          1x,'Solid Density               :',f10.4,' kg/m '/
     c          1x,'Solid Specific Heat         :',f10.4,' J/kg C'/
     c          1x/1x,'INPUT DATA'/1x)
  111  format(1x,'Wall Temperature           :',f9.1,9x,' DegC')
  112  format(1x,'Fusion Temperature         :',f9.1,9x,' DegC')
  113  format(1x,'Initial Liquid Temperature :',f9.1,9x,' DegC')
  114  format(1x,'Latent Heat of Fusion      :',f9.1,9x,' J/kg')
  115  format(1x,'Convective Heat Co-eff.    :',f9.1,9x,' J/m s C')
  116  format(1x,'Initial Bulk Slag Velocity :',f11.3,7x,' m/s')
  117  format(1x,'Bubble Frequency           :',f9.1,9x,' /s')
  118  format(1x,'Inlet Radius               :',f11.3,7x,' m')
  119  format(1x,'Tube Length (slag depth)   :',f11.3,7x,' m')
  501  format(1x,'Wall Elevation (degrees)   :',f10.2)
  502  format(1x,'Distance From Inlet        :',f11.3,7x,' m')
  120  format(1x/1x,'IMMOBILIZED NUMERICAL DATA'/1x/
     c          1x,'28 Radial Segments'/
     c          1x,i2,' Axial Segments'//
     c          1x,'SOLUTION PROFILE'/1x/1x)
  123  format(1x,'INTERFACE PROFILE (mm) AS A FUNCTION OF TIME (s)')
  200  format(1x,i3,f10.4)
  201  format(f16.8)
  202  format(i3)

       return
       end

       subroutine profdata

*************************************************************************
* SUBROUTINE : This routine writes output data produced by plotno 12. *
*************************************************************************

       logical   steady,blocked

       integer*4 nt,tout,axipts

       real*8    t,tmax,pi,dt,dz,var(10),h(0:25),px(0:25),py(0:25),
     c           rad(0:25)

       common    /cons4/pi
       common    /face1/h
       common    /incr2/dz/incr3/dt
       common    /logi1/steady,blocked
       common    /pnts2/axipts
       common    /rads1/rad
       common    /time1/tmax/time2/tout/time4/t/time5/nt
       common    /vars1/var

       if( t.eq.tmax.or.mod(nt,tout).eq.0.or.steady.or.blocked ) then
            if(     blocked) t = t-dt

            if(.not.blocked) then
                 do 1 j = 1,axipts
                 px(j) = ( dble(floatj(j))*dz*dsin(var(10)*pi)
     c                        -(rad(j) - h(j))*dcos(var(10)*pi) )*1000.0
     1           py(j) = 1000.0*( rad(j) - h(j) )*dsin(var(10)*pi)
```

```
            end if

            if(.not.blocked) write(52,100)t
            if(     blocked) write(52,101)t
            if(      steady) write(52,102)
            if(     blocked) go to 4

            j1 = 0
    5       j2 = jmin0( axipts,j1+10 )

            write(52,103)(px(j),j = j1,j2)
            write(52,104)(py(j),j = j1,j2)

            if( j2.lt.axipts ) then
                 j1 = j2 + 1
                 go to 5
            end if

    4       if( .not.blocked.and..not.steady )
     c                          write(72,105)0,axipts
            if(     blocked) write(72,105)1,axipts
            if(      steady) write(72,105)2,axipts
            if(.not.blocked) then
                 write(72,106)t
                 do 6 j = 0,axipts
                 write(72,106)px(j),py(j)
    6            continue
            end if
        end if

  100   format(/1x,'Time =',f10.4,' seconds',$)
  101   format(/1x,'BLOCKAGE for Time >',f10.4,' seconds')
  102   format('+',' - STEADY-STATE')
  103   format(/1x,'Axial Location (mm)  :',11(f8.3,2x))
  104   format(/1x,'Crust Thickness (mm) :',11f10.5)
  105   format(i3/i3)
  106   format(2f24.16)

        return
        end

        subroutine progface

*************************************************************************
* SUBROUTINE : This  routine is  used  for calculating  the interface  *
*              location at successive times. It calculates :-           *
*                1. Temp. gradients at interface                        *
*                2. Interface velocities                                *
*                3. Time step                                           *
*                4. Interface                                           *
*                5. Tests for blockage or steady-state                  *
*************************************************************************

        real*8    h(0:25),dz,dt,t,a2,l(0:28),d1(28),ang,pi,dud1(25),qc,
     c            thdif1,k1,k2,k3,dhdt(25),kts,kt1,dudt,lp,tmax,vflux,
     c            rad(0:25),thdifs,denst1,densts,speht1,speths,var(10)

        integer*4 axipts
```

```
      logical   steady,blocked

      common    /cons2/kts,ktl,qc,lp/cons3/a2/cons4/pi
      common    /facel/h
      common    /fluxl/vflux
      common    /incr2/dz/incr3/dt
      common    /logil/steady,blocked
      common    /phys3/thdifl,thdifs
      common    /phys4/denstl,densts/phys5/spehtl,spehts
      common    /pntsl/l/pnts2/axipts/pnts3/dl
      common    /radsl/rad
      common    /timel/tmax/time4/t/time5/nt
      common    /varsl/var

+-----------------------------------------------------------------------+
| 'RADIAL' TEMPERATURE GRADIENTS AT THE INTERFACE                       |
+-----------------------------------------------------------------------+

      ang = dmod( a2*t,2.0*pi )
      dd  = ( 4.0d0*vflux*(1.0-0.5*dsin(ang)) )/( 9.0*thdifl )

      do 1 j = 1,axipts
    1 dudl(j) = -0.12*( (dd/(j*dz))**(1.0/3.0). )

+-----------------------------------------------------------------------+
| INTERFACE VELOCITIES                                                  |
+-----------------------------------------------------------------------+

      do 2 j = 1,axipts
    2 dhdt(j) = ( kts/(h(j)*dlog(rad(j)/h(j)))
     c            -(ktl*dudl(j))/h(j) + qc )/lp

+-----------------------------------------------------------------------+
| TIME STEP                                                             |
+-----------------------------------------------------------------------+

      jl = 2
      dt = 0.00005d0/dabs( dhdt(1) )
      if( dt.gt.0.4d0 ) then
          dt = 0.4d0
          jl = 1
      end if
      if( var(7).gt.0.0d0 ) then
          if( dt.gt.0.1/var(7) ) dt = 0.1/var(7)
          jl = 1
      end if
      if( t+dt.gt.tmax ) then
          dt = tmax - t
          jl = 1
      end if

+-----------------------------------------------------------------------+
| NEW INTERFACE LOCATION                                                |
+-----------------------------------------------------------------------+

      if( jl.eq.2 ) h(1) = h(1) - 0.00005
      do 5 j = jl,axipts
```

```
    5  h(j) = h(j) + dt*dhdt(j)

+-----------------------------------------------------------------------+
| TEST FOR POSSIBLE BLOCKAGE & STEADY-STATE CONDITIONS                  |
+-----------------------------------------------------------------------+

       steady  = .false.
       blocked = .false.

       do 7 j = 1,axipts
       if( h(j).lt.0.0 ) then
            blocked = .true.
            go to 9
       end if
    7  continue

       do 8 j = 1,axipts
       if( dabs(dhdt(j)).gt.1.0d-12 ) go to 9
    8  continue

       steady = .true.

    9  t  = t + dt
       nt = nt + 1

       return
       end

       subroutine radmeshs

*************************************************************************
* SUBROUTINE : Calculates radial mesh sizes and grid points             *
*************************************************************************

       real*8 l(0:28),dl(28),sum,kk

       common /pnts1/l/pnts3/dl

       dl(1) = 1.0
       do 1 i = 2,28
       sum = 0.0
       do 2 j = 1,i-1
    2  sum = sum + dl(j)
    1  dl(i) = 0.5*(1.0+dsqrt(1.0+4.0d0*sum))
       sum = sum + dl(28)
       kk = 1.0/sum
       l(0) = 0.0
       do 3 i = 1,28
       dl(i) = dl(i)*kk
    3  l(i)  = l(i-1) + dl(i)

       return
       end
```

```
      subroutine readdata

*****************************************************************
* SUBROUTINE : This  routine reads in the required data from the file *
*              from a data file assigned to channel 012.          *
*****************************************************************

      integer*4 plotno,tout,axipts,varnum


      logical   more_data

      real*8    var(10),tmax,varinc,dout,pi,denstl,densts,spehtl,
     c          spehts

      common    /cons4/pi
      common    /doutl/dout
      common    /miscl/more_data
      common    /phys4/denstl,densts/phys5/spehtl,spehts
      common    /pnts2/axipts
      common    /timel/tmax/time2/tout
      common    /varsl/var/vars2/plotno/vars3/varinc/vars4/varnum

      read(12,*)plotno
      do 1 i = 1,10

      read(12,*)var(i)
    1 continue
      read(12,*)tmax,tout
      read(12,*)axipts,dout
      read(12,*)varinc,varnum
      read(12,*)denstl,densts
      read(12,*)spehtl,spehts
      read(12,*)more_data

      pi = 4.0*datan(1.0d0)

      if( plotno.le.10 ) var(plotno) = var(plotno) - varinc

      return
      end

      subroutine starters

*****************************************************************
* SUBROUTINE : This  routine sets up the boundary  and initial states *
*              for the temperature and interface.                 *
*****************************************************************

      integer*4 axipts

      real*8    h(0:25),vflux,dudl(25),t,var(10),u(0:28,0:25),
     c          rad(0:25),lam,thdifl,thdifs,a2

      common    /cons3/a2
      common    /facel/h
      common    /fluxl/vflux
```

```
      common    /incr2/dz
      common    /phys3/thdif1,thdifs
      common    /pnts2/axipts
      common    /rads1/rad
      common    /temp1/u
      common    /time4/t
      common    /vars1/var

      h(0) = var(8)

      call lambda( lam )

      t = (6.25d-10)/(lam*lam*thdifs)

      do 1 j = 1,axipts
    1 h(j) = rad(j) - 0.00005d0

      do 2 j = 0,axipts
      do 2 i = 0,27
    2 u(i,j) = 1.0

      do 3 j = 0,axipts
    3 u(28,j) = 0.0

      return
      end

      subroutine varconst

*****************************************************************************
* SUBROUTINE : This  routine defines  the 'variable' constants  to be *
*              used in the program.                                           *
*****************************************************************************

      integer*4 plotno,axipts,n

      real*8    var(10),dbar,qc,lp,dz,a2,varinc,cota,rad(0:25),pi,
     c          kts,kt1,thconl,thcons,denstl,densts,spehtl,spehts,
     c          thdif1,thdifs

      common    /cons1/dbar/cons2/kts,kt1,qc,lp/cons3/a2/cons4/pi
      common    /incr2/dz
      common    /phys2/thconl,thcons/phys3/thdif1,thdifs
      common    /phys4/denstl,densts/phys5/spehtl,spehts
      common    /pnts2/axipts
      common    /rads1/rad
      common    /vars1/var/vars2/plotno/vars3/varinc

      if( plotno.le.10 ) var(plotno) = var(plotno) + varinc

      cota = dcos(var(10)*pi)/dsin(var(10)*pi)
      dbar = ( var(6)*var(9)*pi
     c        *( 3.0d0*var(8)*var(8)
     c          +3.0d0*var(8)*var(9)*cota
     c          +var(9)*var(9)*cota*cota ) )
     c      /( 3.0d0*var(8)*( (var(8)+var(9)*cota)**3 ) )

      a2 = pi*var(7)
```

```
      dz = var(9)/dble(floatj(axipts))

      do 1 j = 0,axipts + 1
    1 rad(j) = var(8) + dble( floatj(j) )*dz*cota

      thconl = 1.07619949*( -1.0 + (var(3)+var(2))*0.001 )
      thcons = ( 0.96089239/(var(2)-var(1)) )
     c         *( 877.5
     c          - 0.65*var(1)*( 1.0 + 0.00035*var(1) )
     c          - 1.12*var(2)*( 1.0 - 0.001*var(2) ) )

      thdifl = thconl/(denstl*speht1)
      thdifs = thcons/(densts*spehts)

      ktl = thconl*(var(3)-var(2))
      kts = thcons*(var(1)-var(2))
      qc  = var(5)*(var(2)-var(1))
      lp  = var(4)*densts

      return
      end

      subroutine voluflux

*************************************************************************
* SUBROUTINE : This  routine evaluates  the volume flux integral. If    *
*              the number  of sub-intervals is  even (i.e. axipts is    *
*              even)  Simpson's  rule is  applied in composite form.    *
*              If the  number of sub-intervals is odd then Gregory's    *
*              formula is used  (which ensures a similar accuracy as    *
*              Simpson's composite rule).                               *
*************************************************************************

      integer*4 axipts

      real*8    h(0:25),e,vflux,dbar

      common    /consl/dbar
      common    /facel/h
      common    /fluxl/vflux
      common    /pnts2/axipts

      if( jmod(axipts,2).eq.1 ) go to 3

      e = 1.0d0/( h(0)*h(0)*h(0)*h(0) )
     c  + 4.0d0/( h(1)*h(1)*h(1)*h(1) )
      do 1 j = 2,axipts-2,2
    1 e = e
     c  + 2.0d0/( h(j)*h(j)*h(j)*h(j) )
     c  + 4.0d0/( h(j+1)*h(j+1)*h(j+1)*h(j+1) )
      e = e
     c  + 1.0d0/( h(axipts)*h(axipts)*h(axipts)*h(axipts) )
      e = e/( 3.0*dble(floatj(axipts)) )

      vflux = dbar/e

      return
```

```
    3  e =  9.0d0/( h(0)*h(0)*h(0)*h(0) )
     c   + 28.0d0/( h(1)*h(1)*h(1)*h(1) )
     c   + 23.0d0/( h(2)*h(2)*h(2)*h(2) )
        do 2 j = 3,axipts-3
    2  e = e
     c   + 24.0d0/( h(j)*h(j)*h(j)*h(j) )
        e = e
     c   + 23.0d0/( h(j+1)*h(j+1)*h(j+1)*h(j+1) )
     c   + 28.0d0/( h(j+2)*h(j+2)*h(j+2)*h(j+2) )
     c   +  9.0d0/( h(j+3)*h(j+3)*h(j+3)*h(j+3) )
        e = e/( 3.0d0*dble(floatj(axipts)) )

        vflux = dbar/e

        return
        end

        subroutine progkutt

*************************************************************************
* SUBROUTINE : This routine calculates  the temperature distribution *
*              and interface location  at successive times. A Runga- *
*              Kutta integration  scheme is used for the temperature *
*              calculations. The order of evaluation is :-           *
*                1. Temperature gradients at the interface           *
*                2. Interface velocities                             *
*                3. Time step                                        *
*                4. Interface location                               *
*                5. Test for blockage or steady-state interface      *
*                6. Temperature distribution                         *
*************************************************************************

        real*8    h(0:25),u(0:28,0:25),dz,dt,t,a2,l(0:28),dl(28),
     c            dudl(25),vflux,thdifl,cl,dhdt(25),qc,kts,ktl,dudt,
     c            rad(0:25),thdifs,denstl,densts,spehtl,spehts,var(10),
     c            kl,k2,k3,k4,ang,pi,lp,tmax

        integer*4 axipts

        logical   steady,blocked

        common    /cons2/kts,ktl,qc,lp/cons3/a2/cons4/pi
        common    /facel/h
        common    /fluxl/vflux
        common    /incr2/dz/incr3/dt
        common    /logil/steady,blocked
        common    /phys3/thdifl,thdifs
        common    /phys4/denstl,densts/phys5/spehtl,spehts
        common    /pntsl/l/pnts2/axipts/pnts3/dl
        common    /radsl/rad
        common    /templ/u
        common    /timel/tmax/time4/t/time5/nt
        common    /varsl/var
```

'RADIAL' TEMPERATURE GRADIENTS AT THE INTERFACE

```
      ang = dmod( a2*t,2.0*pi )
      dd  = ( 4.0d0*vflux*(1.0-0.5*dsin(ang)) )/( 9.0*thdifl )

      do 1 j = 1,axipts
    1 dudl(j) = -0.12*( (dd/(j*dz))**(1.0/3.0) )
```

```
| INTERFACE VELOCITIES                                                |
```

```
      do 2 j = 1,axipts
    2 dhdt(j) = ( kts/(h(j)*dlog(rad(j)/h(j)))
     c            -(ktl*dudl(j))/h(j) + qc )/lp
```

```
| TIME STEP                                                           |
```

```
      jl = 2
      dt = 0.00005d0/dabs(dhdt(1))
      if( var(7).gt.0.0d0 ) then
           if( dt.gt.0.1d0/var(7) ) then
                dt = 0.1d0/var(7)
                jl = 1
           end if
      end if
      if( dt.gt.0.1d0 ) then
           dt = 0.1d0
           jl = 1
      end if
      if( t+dt.gt.tmax ) then
           dt = tmax - t
           jl = 1
      end if

      t = t + dt
```

```
| INTERFACE LOCATION                                                  |
```

```
      if( jl.eq.2 ) h(1) = h(1) - 0.00005
      do 5 j = jl,axipts
    5 h(j) = h(j) + dt*dhdt(j)
```

```
| TEST FOR BLOCKAGE OR STEADY STATE CONDITIONS                        |
```

```
      steady  = .false.
      blocked = .false.

      do 7 j = 1,axipts
      if( h(j).lt.0.0 ) then
           blocked = .true.
           go to 1000
      end if
    7 continue
```

```
      do 8 j = 1,axipts
      if( dabs(dhdt(j)).gt.1.0d-12 ) go to 9
    8 continue

      steady = .true.

      go to 1000
```

```
+-----------------------------------------------------------------------+
| TEMPERATURE DISTRIBUTION                                              |
+-----------------------------------------------------------------------+
```

```
    9 c1 = thdif1/vflux

      do 11 j = 1,axipts

      do 12 i = 27,1,-1

      k1 = f2( c1,t,u(i,j-1)          ,i,j )
      k2 = f2( c1,t,u(i,j-1)+0.5*dz*k1,i,j )
      k3 = f2( c1,t,u(i,j-1)+0.5*dz*k2,i,j )
      k4 = f2( c1,t,u(i,j-1)+dz*k3    ,i,j )

   12 u(i,j) = u(i,j-1) + ( dz*(k1+k2+k3+k4) )/6.0

      k1 = f1( c1,t,u(0,j-1)          ,j )
      k2 = f1( c1,t,u(0,j-1)+0.5*dz*k1,j )
      k3 = f1( c1,t,u(0,j-1)+0.5*dz*k2,j )
      k4 = f1( c1,t,u(0,j-1)+dz*k3    ,j )

   11 u(0,j) = u(0,j-1) + ( dz*(k1+k2+k3+k4) )/6.0

      nt = nt + 1

 1000 if( t.eq.tmax.or.steady.or.blocked ) then
           write(99,100)tmax
           if( blocked ) write(99,101)t,t-dt
           if( steady  ) write(99,102)t
           if( .not.blocked.and..not.steady ) write(99,103)
           do 333 i = 0,28
  333      write(99,104)(u(i,j),j = 0,20)
      end if

  100 format(1x,'Maximum simulation time is ',f16.8,' seconds'/1x)
  101 format(1x,'Blockage occured between   ',f16.8,' seconds'/
     c       1x,'                       and ',f16.8,' seconds'/1x)
  102 format(1x,'Steady-state conditions at ',f16.8,' seconds'/1x)
  103 format(1x,'Full simulation time completed'/1x)
  104 format(1x/1x,21f6.3)

      return
      end

      double precision function f1( c1,tt,uu,j )

      real*8 c1,tt,uu,u(0:28,0:25),a2,d1(28)
```

```
      common /cons3/a2
      common /pnts3/dl
      common /templ/u

      fl = ( cl*2.0*( u(1,j) - uu ) )
     c     /( dl(1)*dl(1)*( 1.0 - 0.5*dsin(a2*tt) ) )

      return
      end

      double precision function f2( cl,tt,uu,i,j )

      real*8 cl,tt,uu,u(0:28,0:25),a2,dl(28),l(0:28)

      common /cons3/a2
      common /pntsl/l/pnts3/dl
      common /templ/u

      f2 = ( (cl*0.5)/((1.0-l(i)*l(i))*(1.0-0.5*dsin(a2*tt))) )
     c     *( (2.0*((dl(i+1)*u(i-1,j)+dl(i)*u(i+1,j))
     c              /(dl(i)+dl(i+1)) - uu) )/(dl(i)*dl(i+1))
     c       +(u(i+1,j)-u(i-1,j))/(dl(i+1)+dl(i)) )

      return
      end
```

REFERENCES

ABRAMOWITZ M. and STEGUN I.A., Handbook of Mathematical Functions, Dover Publications, New York (1964)

ABUAF N. and GUTFINGER C., 'Heat transfer with a moving boundary: Application to fluidized-bed coating of thin plates', Int. J. Heat Mass Transfer 16, 213-216 (1973)

ATTHEY D.R., 'A finite difference scheme for melting problems', J. Inst. Maths. Applics. 13, 353-366 (1974)

BELL G.E., 'On the performance of the enthalpy method', Int. J. Heat Mass Transfer 25, 587-589 (1982)

BELL G.E. and CRANK J., 'A method of treating boundary singularities in time-dependent problems', J. Inst. Maths. Applics. 12, 37-48 (1973)

BELL G.E. and RITCHIE S.I.M., 'The use of the odd-even hopscotch algorithm for a thermal shock problem', J. Comp. Phys. 38, 413-419 (1980)

BELL G.E. and WOOD A.S., 'On the performance of the enthalpy method in the region of a singularity', Univ. St Andrews Preprint (July 1982)

BIRTWHISTLE G., The Principles of Thermodynamics, 2nd Edn., Clarendon Press, Oxford (1927)

BOYCE W.E. and DIPRIMA R.C., Elementary Differential Equations and Boundary Value Problems, 3rd Edn., John Wiley & Sons Ltd, New york (1965)

CARSLAW H.S., Introduction to the Mathematical Theory of the Conduction of Heat in Solids, 2nd Edn., Macmillan, London (1921)

CARSLAW H.S. and JAEGER J.C., Conduction of Heat in Solids, 2nd Edn., Clarendon Press, Oxford (1959)

CHERNOUS'KO F.L., 'Solution of non-linear heat conduction problems in media with phase change', Int. Chem. Engineering 10, No. 1, 42-48 (1970) [First published (Russian) in Zh. Prikl. Mekh. Tekh. Fiz., No. 2, 6-14 (1969)]

CHURCHILL R.V., Complex Variables and Applications, 2nd Edn., McGraw-Hill, New York (1960)

CRANK J., 'Two methods for the numerical solution of moving boundary problems in diffusion and heat flow', Quart. J. Mech. & App. Maths. 10, 220-231 (1957)

CRANK J., 'Finite difference methods' in Ockendon & Hogkins, pp.192-207 (1975)$_1$

CRANK J., 'A nostalgic look at the mechanical solution of Stefan problems', Bull. I.M.A. 11, 32-33 (1975)$_2$

CRANK J. and PHAHLE R.D., 'Melting ice by the isotherm migration method', Bull. I.M.A. 9, 12-14 (1973)

DIX R.C. and CIZEK J., 'The isotherm migration method for transient heat conduction analysis' in Proceedings of the Fourth International Heat Transfer Conference, Paris 1 ASME, New York (1971)

DOETSCH G., Guide to the Applications of Laplace Transforms, Van Nostrand, Munich (1961)

DUDA J.L., MALONE M.F., NOTTER R.H. and VRENTAS J.S., 'Analysis of two-dimensional diffusion-controlled moving boundary problems', Int. J. Heat Mass Transfer 18, 901-910 (1975)

EHRICH O., CHUANG YUN-KEN and SCHWERDTFEGER K., 'The melting of metal spheres involving the initially frozen shells with different material properties', Int. J. Heat Mass Transfer 21, 341-349 (1978)

FERRISS D.H., 'Fixation of a moving boundary by means of a change of independent variable' in Ockendon & Hogkins, pp.251-255 (1975)

FOX L. and SANKAR R., 'Boundary singularities in linear elliptic differential equations', J. Inst. Maths. Applics. 5, 340-350 (1969)

GOLDSTEIN M.E. and SIEGEL R., 'Conformal mapping for heat conduction in a region with an unknown boundary', Int. J. Heat Mass Transfer 13, 1632-1636 (1970)

GOURLAY A.R., 'Hopscotch: a fast second-order partial differential equation solver', J. Inst. Maths. Applics. 6, 375-390 (1970)

GRAETZ L., ' ', Ann. Phys. u. Chem. 25, 337- (1885)

HOWARTH L.(Ed.), Modern Developments in Fluid Dynamics - High Speed Flow, Vol.II, Clarendon Press, Oxford (1953)

HOY H.R., ROBERTS A.G. and WILKINS G.M., 'Behaviour of mineral matter in slagging gasification processes', I.G.E.J. comm.672, 444-469 (1965)

LANDAU H.G., 'Heat conduction in a melting solid', Quart. J. App. Maths. 8, 81-94 (1950)

LIGHTFOOT N.M.H., 'The solidification of molten steel', Proc. London Math. Soc., (2) 31, 97-116 (1930)

MASSEY W.N. and SUNDERLAND J.E., 'Heat and mass transfer in semi-porous channels with application to freeze-drying', Int. J. Heat Mass Transfer 15, 493-502 (1972)

MOALEM D., SIDEMAN S., ORELL A. and HETSRONI G., 'Direct contact heat transfer with change of phase: Condensation of a bubble train', Int. J. Heat Mass Transfer 16, 2305-2319 (1973)

MOTZ H., 'The treatment of singularities of partial differential equations by relaxation methods', Quart. J. Appl. Maths. 4, 371-377 (1946)

MURRAY W.D and LANDIS F., 'Numerical and machine solutions of the transient heat conduction problems invlolving melting or freezing', J. Heat Transfer 81C, 106-112 (1959)

McLACHLAN N.W., 'Heat conduction in elliptical cylinder and an analogous electromagnetic problem', Phil. Mag., (7) 36, 600-609 (1945)

McLACHLAN N.W., Theory and Application of Mathieu Functions, Clarendon Press, Oxford (1947)

NEUMANN F., c.f. Riemann-Weber, Die partiellen differentialgleichungen der mathematischen physik, edn. 5 2, p.121 (1912)

OCKENDON J.R. and HOGKINS W.R., Moving Boundary Problems in Heat Flow and Diffusion (Eds.), 2nd edn., Clarendon Press, Oxford (1975)

OZISIK M.N. and MULLIGAN J.C., 'Transient freezing of liquids in forced flow inside circular tubes', ASME J. Heat Transfer 91(C), 385-389 (1969)

PATEL P.D., 'Interface conditions in heat conduction problems with change of phase', AIAA J. 6, 2454 (1968)

PHILLIPS G.M. and TAYLOR P.J., Theory and Applications of Numerical Analysis, Academic Press, London (1973)

RATHJEN K.A. and JIJI L.M., 'Heat conduction with melting or freezing in a corner', ASME J. Heat Transfer, 91C, 101-109 (1971)

ROSE M.E., A method for calculating solutions of parabolic equations with a free boundary', Maths. Comp. 14, 249-256 (1960)

RUISSEAUX N. Des and ZERKLE R.D., 'Freezing of hydraulic systems', Can. J. Chem. Engineering 47, 233-237 (19699

SAITO T., 'On the distribution of temperature in steel ingots during cooling', Tohoku Imperial Univ. Science Reports, No.10 (1921)

SAMPSON P. and GIBSON R.D., 'A mathematical model of nozzle blockage by freezing', Int. J. Heat Mass Transfer 24, 231-241 (1981)

SAMPSON P. and GIBSON R.D., 'A mathematical model of nozzle blockage by freezing - II. Turbulent flow', Int. J. Heat Mass Transfer 25, 119-126 (1982)

SIEGEL R., 'Conformal mapping for steady two-dimensional solidification on a cold surface in a flowing liquid', NASA Tech. Note D-4771 (1968)

SIEGEL R., 'Analysis of solidification interface shape during continuous casting of a slab', Int. J. Heat Mass Transfer 21, 1421-1430 (1978)

SMITH G.D., Numerical Solution of Partial Differential Equations: Finite Difference Methods, 2nd Edn., Clarendon Press, Oxford (1978)

STEFAN, 'Uber die Theorie der Eisbildung inbesondere uber die Eisbildung im Polarmeere', Ann. Phys. u. Chem. 42, 269-286 (1891)

SZEKELY J. and LEE R.G., 'The effect of slag thickness on heat loss from ladles holding molten steel', Trans. A.I.M.E. 242, 961-965 (1968)

SZEKELY T. and THEMELIS N.J., Rate Phenomena in Process Metallurgy, Wiley-Inerscience (1971)

TART K.R. and RAMPLING T.W.A., 'Fixed bed slagging gasification - A

means of producing synthetic fuels and feedstocks', British Gas Report MRS E 376 (Presented at Coal-Chem 2000 conference, sept.1980)

VOLLER V. and CROSS M., 'Accurate solutions of moving boundary problems using the enthalpy method', Int. J. Heat Mass Transfer 24, 545-556 (1981)

WATSON G.N., A Treatise on the Theory of Bessel Functions, 2nd Edn., CUP, Cambridge (1944)

WOOD A.S., RITCHIE S.I.M. and BELL G.E., 'An efficient implementation of the enthalpy method', Int. J. Num. Meths. Engineering 17, 301-305 (1981)

YOSHIKAWA F. and SZEKELY J., 'Mechanism of blast furnace erosion', Ironmaking and Steelmaking No.4, 159-168 (1981)

ZERKLE R.D. and SUNDERLAND J.E., 'The effect of liquid solidification in a tube upon the laminar heat transfer and pressure drop', ASME J. Heat Transfer 90(C), 183-190 (1968)